VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ŘÍZENÍ PROJEKTŮ V IT FIRMĚ

INFORMATION SYSTEM FOR PROJECT MANAGEMENT IN AN IT COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ZDENĚK KAMENSKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21224/2017/xkamen06

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Kamenský Zdeněk**
Obor: Informační technologie
Téma: **Informační systém pro řízení projektů v IT firmě**
Information System for Project Management in an IT Company
Kategorie: Informační systémy

Pokyny:

1. Senzamte se s principy tvorby webových aplikací, zaměřte se na jazyk PHP a framework Laravel.
2. Analyzujte požadavky na informační systém pro řízení projektů zahrnující evidenci projektů, úkolů, týmů řešících projekty, možnost nahrávat soubory, vkládat komentáře k projektům atd.
3. Navrhněte informační systém splňující tyto požadavky. Zaměřte se na rychlost práce s informačním systémem (cachování výsledků dotazů, nahrazení stávajícího řešení databázového dotazování v Laravelu rychlejší řešením ...) a na jednoduchost uživatelského rozhraní.
4. Navržený informační systém implementujte a ověřte jeho funkčnost.
5. Porovnejte vaše řešení s existujícím řešením podobného zaměření z hlediska rychlosti dotazování.
6. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této bakalářské práce je navrhnout a implementovat webový informační systém pro řízení projektů v IT firmě. Uživatelé mohou po přihlášení do systému zakládat týmy a projekty. V každém týmu mohou jeho členové přidávat týmové příspěvky s různým typem informací a komentovat je. Do projektů mohou jejich řešitelé nahrát soubory se zadáním práce, nebo soubory obsahující výsledky dané práce. Řešitelé projektu smí také přidávat nové úkoly, které je nutné v projektu řešit. V systému je kladen důraz na rychlost získávání dat z databáze. Při vývoji byl použit PHP framework Laravel verze 5.5.

Abstract

The goal of this bachelor thesis is to design and implement the web information system for managing of the projects in an IT company. Logged users are allowed to create teams and establish new projects. Members of each team can add team articles with various type of information and comment them. Users, who are assigned to the particular project can upload files with the tasks or files with solution. They can also add new tasks for each project which are needed to be solved. Quick data feed from database is accented in this information system. PHP framework Laravel version 5.5. has been used during the development of the system.

Klíčová slova

Informační systém, řízení projektů, webová aplikace, Laravel, PHP, MYSQL, HTML, CSS, Bootstrap

Keywords

Information system, project management, web application, Laravel, PHP, MYSQL, HTML, CSS, Bootstrap

Citace

KAMENSKÝ, Zdeněk. *Informační systém pro řízení projektů v IT firmě*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém pro řízení projektů v IT firmě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Zdeněk Kamenský
14. května 2018

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Vladimíru Bartíkovi, Ph.D. za vedení při tvorbě této bakalářské práce, dále pak své přítelkyni a rodině za podporu.

Obsah

Kapitola 1

Úvod

Hlavním cílem této bakalářské práce je navrhnout a realizovat informační systém pro řízení projektů v IT firmě. Kdo někdy pracoval ve středně velké, nebo dokonce velké IT firmě, tak dobře ví, jak důležité je řízení projektů. Taková firma se většinou nezabývá jedním projektem, nebo úkolem, nýbrž její zaměstnanci realizují velké množství nejrůznějších projektů zároveň. Tvoří se mnohačlenné týmy, které se buď specializují na určitou problematiku, zakázku, zákazníka, nebo jsou k dispozici jako IT podpora. Členové týmů mezi sebou potřebují sdílet různé informace, jako třeba čas a místo konání další schůzky, nebo obecné pokyny od klientů. Důležitá je také možnost interakce, třeba pomocí komentářů. U projektů je podstatné rozdělení jednotlivých částí realizace mezi více osob. Vzhledem k netrivialitě projektů bývá zvykem, že se na nich podílí více odborníků zaměřených na tu část, které rozumí. Proto je nutné práci zorganizovat, stanovit dílčí cíle a přiřadit je kompetentním osobám. Výše uvedené může mnohonásobně ulehčit dobře navržený informační systém zaměřený na řízení projektů.

V dnešní době je možné se nejčastěji setkat se dvěma typy informačních systémů. Jedním z nich je desktopový informační systém, fungující na principu klient-server, který je fyzicky instalován přímo do zařízení zaměstnanců a je tak dostupný pouze z těchto zařízení. Propojení takového systému bývá realizováno pomocí firemní sítě, včetně úložiště na některém z firemních serverů. Druhým typem je webový informační systém, který běží na webovém serveru a uživatelé se k němu připojují odkudkoliv pomocí webových prohlížečů. Výhodou tohoto systému je dostupnost. Uživateli stačí mít pouze připojení k internetu a znát svoje přihlašovací údaje.

Webový informační systém se od systému fungujícího na principu klient-server výrazně liší. Odlišný je jak způsob návrhu jednolivých částí systému, tak využité techniky implementace a zejména pak použité technologie. Tato bakalářská práce je zaměřena právě na webový informační systém určený, jak již bylo uvedeno výše, pro řízení projektů v IT firmě s důrazem na rychlost získávání dat z databáze.

V následujících kapitolách budou nejprve vysvětleny a popsány základní pojmy týkající se této problematiky. Následně budou objasněny technologie použité při návrhu a realizaci systému. Stručně bude uveden také verzovací systém Git. Následně pak bude vysvětlen samotný návrh celého systému včetně různých diagramů, ať už popisujících funkčnost systému, či strukturu databáze. Poté bude popsána vlastní implementace, kde bude podrobněji vysvětlen postup při snaze co nejvíce zrychlit získávání dat z databáze a tím přispět ke zrychlení odezvy systému jako celku a představeno bude vlastní řešení. Proběhne také porovnání s existujícím řešením přímo od vývojářů frameworku Laravel. Dále proběhne

testování a měření a následné zhodnocení dosažených výsledků. Na závěr budou uvedena možná budoucí rozšíření.

Kapitola 2

Základní pojmy

V následujících řádcích budou objasněny některé základní pojmy, které je nutné bezpečně znát při tvorbě informačních systémů a zejména pak při čtení této práce. Každý pojem bude samostatně stručně popsán.

2.1 Informace

V této sekci je vysvětlen pojem informace. Nejprve je tento pojem popsán obecně, poté je objasněn pojem data a náledně informace právě ve vztahu k datům.

2.1.1 Informace obecně

Na pojem informace je možné nahlížet z různých úhlů pohledu. Pokud vycházíme z všedního neformálního pohledu, pak informace může být chápána jako zpráva nebo jazykový projev, ve kterém se konstatují různá fakta. Naopak z filosofického hlediska se pod označením informace rozumí vnímatelný obsah poznaného nebo předpokládaného obrazu skutečnosti, který je možno využít pro život člověka, nebo trochu konkrétněji představuje míru uspořádanosti systémů. Pojem informace lze také vykládat z hlediska přírodních věd či kybernetiky.

Nicméně pro účely této práce a obecně informačních systémů jsou uvedené definice nepřesné nebo až příliš obecné. Pro účely této práce jsou pojmy informace a s nimi související data popsány v následujících odstavcích. [17]

2.1.2 Data

Data je možné chápat jako určitou prezentaci skutečnosti, která je zároveň schopná přenosu, uchování, zpracování nebo interpretace. Z hlediska informatiky a počítačů jsou data položky, které mají nějakou hodnotu a jsou různých datových typů. Data nemusejí mít a většinou nemají sémantiku a jsou zpracovány čistě formálně na základě syntaxe, jelikož je lze pomerně snadno syntakticky definovat. Hodnoty dat z pravidla udávají stav systému. [17]

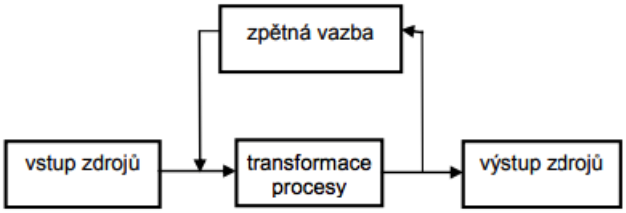
2.1.3 Informace ve vztahu k datům

Informace je možné chápat jako data, která mají sémantiku. Definice sémantiky, na rozdíl od definice syntaxe, je poměrně náročná záležitost. Buď je dána zavedenými konvencemi všech uživatelů dané informace, nebo je popisována slovně, neformálně a nepřesně. Označení informační systémy může být značně matoucí, jelikož jde v podstatě vždy v zásadě

o zpracování dat, nikoliv informací. Data se informacemi stávají až po vhodné interpretaci uživatelem. Každý z uživelů může daná data interpretovat různým způsobem. Často tedy dva uživatelé extrahují z totožných dat různé informace, a tím mohou vznikat různé kolize a problémy. Je proto třeba zajistit, aby u většiny uživatelů existovala shodná interpretace dat, například s využitím prostředků ontologie. [17]

2.2 Systém

Systém obecně je množnina prvků a vazeb mezi nimi, které jsou definovány za určitým účelem na nějakém nosiči. Přičemž nosičem může být množina prvků daného systému ve vzájemných informačních a procesních vztazích. Prvky nosičů se označují jako zdroje. Na následujícím obrázku se nachází obecné schéma systému. Jak je možné vidět, do systému vstupují určité vstupní zdroje, probíhá zde transformace vstupních zdrojů na výstupní a také procesy. Další součástí systému je zpětná vazba, která udává stav sytému. Nosičem v systému zpravidla bývá jakási část reálného světa, jako například firma, škola a podobně. Typickými zdroji v nosičích jsou studenti, výrobní stroje a automobily. Těmto zdrojům se říká fyzické a tvoří takzvané fyzické systémy. Dalším možným zdrojem jsou informace a tvoří informační systémy. Dále se také systémy dělí na otevřené, interagující se svým okolím a naopak uzavřené. [17]

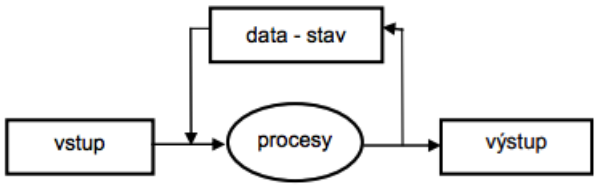


Obrázek 2.1: Obecné schéma systému [17]

2.3 Informační systém

Informační systém je obecný systém pracující s konceptuálními zdroji, tedy informacemi. Jak je možné vidět na následujícím schématu, část transformační a část se zpětnou vazbou lze pojmenovat konkrétněji. Do informačního systému se vkládají vstupní informace a následně po transformaci získávají informace výstupní. V transformační části probíhají různé procesy a nad informacemi jsou vykonávány algoritmy. Podobně jako v obecném systému je i zde část, která udává stav systému. Ne všechny výstupy procesů a transformací jsou tedy závislé pouze na okamžitých vstupních datech, ale také na aktuálním stavu systému.

Informační systém není autonomní, nýbrž modeluje nějaký, buď reálný, nebo nereálný fyzický systém. Bývá zpravidla otevřeným systémem, kterému jsou informace dodávány na vstupu a odebírány na výstupu. Informační systém nikdy nemodeluje daný fyzický systém 1:1, ale existuje zde vždy jistá míra abstrakce. [17]



Obrázek 2.2: Schéma informačního systému [17]

2.4 Databáze

Je systém uchovávající perzistentní data. Do pojmu databáze se řadí i softwarové nástroje určené k přístupu k uloženým datům a k manipulaci s nimi. Tento software bývá označován jako systém řízení báze dat.

Existuje několik základních typů databází klasifikovaných dle vazeb mezi entitami, či způsobu ukládání dat, jako například objekové, hierarchické, nebo relační. V této práci bude využíváno právě relační databáze.

2.4.1 Relační databáze

Relační databáze definuje soubory, či tabulky jako relace.

Relační databázi je možné si představit jako například kartotéku u obvodního lékaře. Každý pacient má zde svojí kartu - relaci (tabulku). O pacientech jsou uchovány informace, jako například jméno, příjmení, datum narození nebo bydliště. Tyto údaje se v relačních databázích označují souhrně jako atributy. Každý atribut může nabývat hodnot z nějaké množiny přípustných hodnot. Tyto obory hodnot atributů se označují jako domény. Informace o jednom pacientovi vytváří vzájemný vztah konkrétních hodnot z příslušných domén. Tyto údaje, které jsou spolu ve vztahu (definují jednoho konkrétního pacienta) se v relačním modelu označnují jako n-tice. Taková tabulka obsahuje v uvedeném příkladě několik řádků, každý pro jednoho pacienta. Tabulka bude také obsahovat sloupce. Každý sloupec bude odpovídat jednomu atributu, například jeden sloupec pro jméno, další pro příjmení a tak dále (viz následující obrázek). [22]

Toto byl krátký úvod do problematiky databází, zejména pak relačních databází, které jsou využívány v této bakalářské práci.

LOGIN	JMÉNO	PŘÍJMENÍ	ADRESA
xcerny00	Petr	Černý	Brněnská 15 Vyškov
xnovak00	Jan	Novák	Cejl 9 Brno
xnovak01	Pavel	Novák	Cejl 9 Brno

Obrázek 2.3: Relace znázorněná tabulkou [22]

2.5 Frontend

Zjednodušeně je frontend to, co nějaký uživatel, popřípadě externí systém vidí a s čím může interagovat. Například v případě webového informačního systému uživatel vidí tabulku všech registrovaných uživatelů. Nemá zde informace o tom, odkud byla data extrahována, popřípadě jakými způsoby byla upravena. Uživatel vidí pouze vhodně prezentovaný výsledek ve formě tabulky realizovaný například pomocí HTML a CSS.

2.6 Backend

Dalo by se říct, že backend je pravý opak frontendu, nicméně spolu velmi úzce souvisí. Backend bývá zpravidla běžným uživatelům skryt. Určuje data, která budou prezentována frontendem. Opět například u webového informačního systému vidí uživatel nějakou tabulku. Backend představuje to, jakým způsobem byla data zobrazená v tabulce, vybrána z databáze, souboru, nebo jiného zdroje a jak tato data byla upravena.

2.7 Uživatelské rozhraní

Uživatelsky orientované rozhraní je takové rozhraní, které je primárně orientované na uživatele a samotné prvky rozhraní nemusí odpovídat přímo funkčním částem systému nebo zařízení. Uživatel tedy nemusí rozumět principu funkcí daného systému či zařízení. Tato rozhraní se snaží co nejvíce odstínit uživatele od toho, jak to v dané aplikaci funguje, že jsou aktuálně zpracovány nějaké řetězce, nebo že právě probíhá dotazování systému do databáze. Uživateli je toto skryto a naopak jsou mu v přívětivé formě prezentovány výsledky průběhu procesů dané aplikace. Tím zůstavají skryty složité programové konstrukce a tedy odpadá nutnost jejich porozumění. [21]

2.8 Objektově orientované programování

Objektově orientované programování (dále jen OOP) je přístup, kde jsou vytvářeny softwarové objekty korespondující s objekty fyzickými. Jinými slovy řečeno je OOP způsob programování různých aplikací, kde jsou vytvářeny objekty, které do jisté míry odpovídají objektům z reálného světa, které jsou jejich předlohou. Mezi reálnými objekty a těmi softwarovými existuje určitá míra abstrakce. Výhodou OOP je, že softwarový a reálný model jsou analogické. Softwarové modely jsou také flexibilní a znovupoužitelné. Součástí OOP je analýza a specifikace požadavků, návrh, implementace a případně také testování. Základní koncepty OOP jsou:

- objekt,
- abstrakce,
- zapouzdření,
- dědičnost a
- mnohotvárnost. [18]

2.8.1 Objekt

Jak vyplývá z předchozího odstavce, základním stavebním kamenem OOP je objekt. Ten uchovává a sjednocuje data a funkcionalitu. Je tedy možné nějaký problém patřičně rozdělit na menší podproblémy, které mohou být reprezentovány pomocí objektů. [18]

2.8.2 Abstrakce

Z určitého pohledu se pak může objekt tvářit jako jakási černá skříňka (black box), která má nějakou funkcionalitu, ale její vnitřní realizace je okolí skryta. Dále existuje také určitá míra abstrakce mezi softwarovým a reálným objektem, který je mu předlohou. Pro potřeby dané aplikace či systému mohou zůstat některé detaily skryty, či zanedbány. Programátor volí vhodou úroveň abstrakce pro daný problém, či objekt. [18]

2.8.3 Zapouzdření

Jak objekty reálného světa (například lidé), tak i softwarové objekty spolu určitým způsobem komunikují. Softwarové objekty spolu komunikují vždy prostřednictvím rozhraní. Rozhraní definuje jaké operace mohou být nad daným objektem realizovány. Protokol pak udává přesný způsob komunikace mezi objekty. Ostatní objekty tedy při komunikaci využívají toto rozhraní, přičemž skutečná implentace objektu či jeho metod je plně skryta. [18]

2.8.4 Dědičnost

Nově vytvořené objekty mohou sdílet a rozšiřovat chování již existujících objektů. Tato vlastnost slouží ke sdílení implementace, nebo ke specializaci některých objektů. Objekty v takovém případě můžeme označovat jako mateřské (poskytují své chování či vlastnosti) a dceřiné (rozšiřují chování či vlastnosti). Dědičnost je nedílnou součástí OOP a je důležitá pro udržovatelnost a rozšiřitelnost objektově orientovaných systémů. [18]

2.8.5 Mnohotvárnost

Mnohotvárnost využívá principu zasílání zpráv objektům v OOP. Různým objektům jsou zaslány zprávy, přičemž reakce na ně závisí na konkrétním objektu. To znamená, že dvěma různým objektům může být zaslána stejná zpráva, avšak reakce se mohou různit podle jejich konkrétní implementace. [18]

Framework Laravel, použitý v této bakalářské práci je založený na jazyce PHP a objektově orientovaném programování.

Kapitola 3

Použité technologie

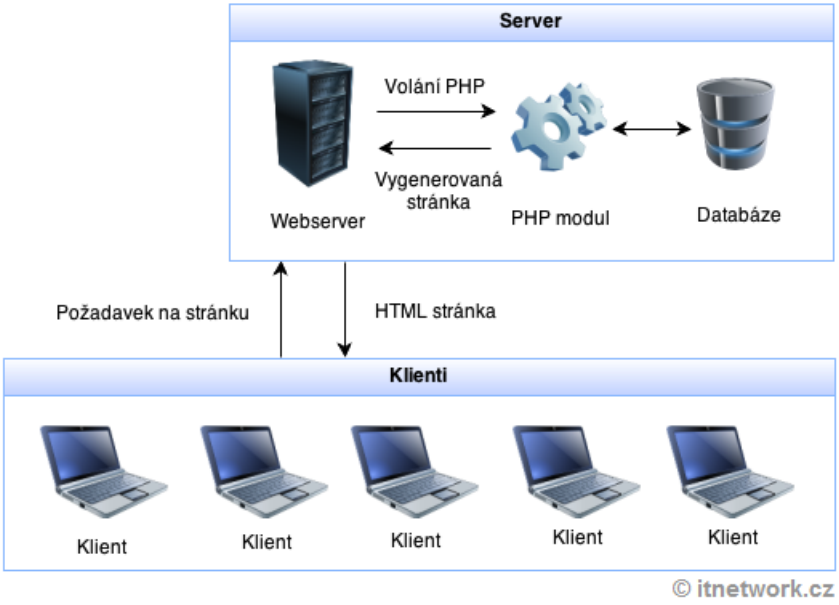
V této kapitole jsou stručně popsány technologie, které byly využity při tvorbě této bakalářské práce. Celá aplikace je vytvořena v open source PHP frameworku Laravel. Z toho vyplývá, že takřka celý backend aplikace je napsán v programovacím jazyce PHP. Struktura webových dokumentů je v jazyce HTML a podrobnější vzhled jednotlivých webových stránek aplikace je definován pomocí kaskádových stylů, tedy CSS. Pro jednoduchou kombinaci HTML kódu s prvky jazyka PHP (například řídící struktury) byly použity Blade šablony frameworku Laravel. Součástí tohoto frameworku je také CSS framework Bootstrap. Pro práci s databází, ať už připojení, vkládání dat, upravování dat, nebo získávání dat z různých tabulek byl použit jazyk MySQL spolu s nástrojem Query Builder. Správa verzí probíhala pomocí verzovacího systému Git. Návrh celého informačního systému byl vytvořen v modelovacím jazyce UML.

3.1 PHP

Český význam zkratky PHP je Hypertextový preprocesor. Původní význam byl Personal Home Page. Tyto zkratky mohou být trochu matoucí, nicméně výraz PHP reprezentuje open source programovací jazyk řadící se do rodiny skriptovacích jazyků. Využití může najít pro tvorbu konzolových či desktopových aplikací, ať už v kompilované (překládané) nebo interpretované formě. Hlavní využití je však pro vývoj dynamických webových aplikací, kdy veškeré vykonávání programu probíhá na straně serveru, ke kterému se uživatelé připojují pomocí klientských aplikací, například webových prohlížečů. Výhodou PHP je možnost vložení funkčního kódu přímo do HTML. Na rozdíl například od Javascriptu na straně klienta, je kód PHP spuštěn na straně serveru a generuje kód HTML, který je odeslán na stranu klienta, kde je ve vhodné formě prezentován. Významnou funkcí je také propojení s databází uloženou na straně serveru.

Aktuální verze programovacího jazyka PHP, která je použita v této bakalářské práci je 7.1. Na rozdíl od předchozích verzí umožňuje tato verze například přímo deklarovat datové typy parametrů metod, deklarovat návratové datové typy metod, včetně návratového typu void, datové typu nullable, které akceptují jak daný datový typ, tak také hodnotu null, záporné offsety ve funkcích pro práci s řetězci, nebo odchytávání více typů vyjímek v jedné struktuře catch a další. Tato verze PHP je také plně kompatabilní s použitým frameworkem Laravel verze 5.5.

Při realizaci této bakalářské práce se vycházelo z oficiálních stránek jazyka PHP a z na těchto stránkách dostupných manuálů a dokumentace. [10] [16] [19]



Obrázek 3.1: Schéma webového serveru využívajícího jazyk PHP [5]

3.2 HTML

HTML je zkratka pro HyperText Markup Language. Jedná se tedy, jak překlad napovídá, o hypertextový značkovací jazyk, který vyvinul Tim Berners-Lee v roce 1990. Používá se pro vytváření elektronických dokumentů, zobrazovaných na World Wide Web. Tyto dokumenty tvoří webové stránky. Jednotlivé stránky mohou být vzájemně propojeny pomocí hypertextových odkazů. Ať navštívíte jakoukoliv webovou stránku, vždy je napsána v nějaké verzi HTML kódu. Pomocí HTML kódu je možné zajistit správné formátování textu, tabulek či obrázků tak, aby byly zobrazeny internetovým prohlížečem, jak je požadováno. HTML poskytuje základní strukturu stránky, o podrobnější vzhled či chování jednotlivých prvků se starají kaskádové styly (viz následující sekce), ať už vložené přímo do HTML kódu, či oddělené v samostatném souboru. HTML tedy představuje více méně strukturu webové stránky.

Pro vývojáře či návrháře webových stránek je klíčové pamatovat si, nebo umět vyhledat takzvané značky (tags), ze kterých se jazyk HTML skládá. Jejich použití pak bývá většinou triviální. Rozlišují se značky řádkové či blokové a dají se podle určitých pravidel různě zanořovat. [4]

3.2.1 HTML5

HTML5 je nejnovější verze značkovacího jazyka HTML vytvořená aktualizací předchozí verze, tedy HTML4. HTML5 používá stejná základní pravidla a principy jako HTML4, avšak přidává některé nové značky, elementy či atributy, které umožňují lépe pracovat se sémantikou prvků a také umožňují použití dynamických prvků, které jsou aktivovány pomocí jazyka JavaScript. [4]

Tato verze HTML, tedy HTML5 je použita v této bakalářské práci.

3.3 CSS - Kaskádové styly

Jak již bylo uvedeno v předchozím odstavci, elementy jazyka HTML mají jistou sémantiku, pomocí které sestaví uvedené informace v dokumentu. O zobrazení se pak stará prohlížeč. S postupným rozšiřováním HTML se zvyšovala potřeba ovlivnit podrobněji vzhled jednotlivých prvků v dokumentu, či jejich další chování. Postupně proto vznikly kaskádové styly, tedy CSS (Cascading Style Sheets). Styl každého prvku určuje, jakým způsobem bude daný prvek zobrazen a jaké bude mít další chování. Typickými příkady jsou barva, umístění prvku, nebo jeho rozměry. Tyto a další vlastnosti jsou definovány pomocí pravidel, tedy stylů. Slovo kaskádové značí, že jednotlivá pravidla jsou intepretována v kaskádě a to v pořadí: externí šablony stylů, lokální styly dokumentu a lokální styl konkrétního prvku. V předchozích verzích HTML a CSS byly často styly vkládány přímo do značek jazyka HTML v dokumentu. S potřebou refaktorizace a různých úprav vzhledu se tyto styly postupně oddělovaly do speciálních souborů. V původním HTML dokumentu je pak uveden odkaz na tento soubor. Jednotlivé elementy HTML dokumentu jsou rozlišovány pomocí identifikátorů a tříd. Aktuální používaná verze CSS je verze 3. [15]

Ve frameworku Laravel je použito rozšíření Sass jazyka CSS umožňující vytvářet proměnné a zanořovat jednotlivé elementy. Jedná se o skriptovací jazyk, který je interpretován, nebo v případě tohoto frameworku kompilován pomocí programu npm [9] do CSS. [12]

Při tvorbě této bakalářské práce se vychází ze znalostí nabytých na webových stránkách W3Schools [13] a Jak psát web [6].

3.4 MySQL

Databázový server MySQL patří k nejrozšířenějším open source databázovým serverům. Nejčastěji, avšak není to pravidlem, se MySQL používá v kombinaci s PHP pro tvorbu dynamických webových stránek či aplikací. Dále také vyniká svojí celkovou jednoduchostí a rychlostí. MySQL server je implicitně využíván jako databázový server pro aplikace vytvářené ve frameworku Laravel a byl tedy použit při tvorbě této bakalářské práce. [16]

3.5 Laravel

Laravel je nejvíce rozšířený PHP framework na světě. Pro tuto práci byla vybrána jeho verze 5.5, která byla vydána 30. srpna 2017. Laravel poskytuje velmi přehledné a rychlé řešení pro vývoj webových aplikací. Je rozšířením architektury MVC, tedy model - view - controller, avšak je postaven na mnoha dalších návrhových vzorech jako například DRY, KISS, SOLID, Factory, Facade nebo Provider. V této práci byl také využit návrhový vzor Repository.

Laravel si zakládá na přehlednosti, rychlosti, udržovatelnosti kódu a také jeho čistotě. Tento framework také zprostředkovává připojení k databázi a dalším prvkům. Laravel využívá právě pro práci s databází prostředky jako Query Builder, migrace, seedy, či vlastní ORM Eloquent, který bude v této bakalářské práci nahrazen vlastním řešením. Mimo jiné Laravel využívá Blade šablony pro tvorbu frontendu a také CSS rozšíření Sass a software npm mimo jiné právě pro kompilaci Sass do CSS. Laravel nabízí také snadnou konfiguraci vytvořených aplikací pomocí konfiguračních souborů a proměnných prostředí (enviromental). Nedílnou součástí tohoto frameworku je Composer sloužící pro správu závislostí projektu. [7]

3.5.1 Query Builder

Query Builder je nástroj frameworku Laravel, který poskytuje jednoduché a velmi srozumitelné rozhraní pro práci s databází. Umožňuje jak základní operace nad databází, jako jsou získávání dat z databáze, úprava nebo vkládání, tak poskytuje také složitější konstrukce, jako například různé typy podmíněného dotazování (where klauzule), řazení dat (například orderBy) či jejich seskupování (groupBy). Query Builder využívá vazbu parametrů PDO (PHP Data Objects) k ochraně aplikace před různými SQL útoky. Příklad použití Query Builderu je zobrazen na následujícím obrázku (**3.2**). [8]

Databázové rozhraní Query Builder je použito i v této bakalářské práci.

```
DB::table('users')
            ->where('name', '=', 'John')
            ->where(function ($query) {
                $query->where('votes', '>', 100)
                      ->orWhere('title', '=', 'Admin');
            })
            ->get();
```

Obrázek 3.2: Příklad použití Query Builderu [8]

3.5.2 Blade

Jedná se o jednoduchý šablonovací nástroj přímo od vývojářů frameworku Laravel. Díky němu je velmi snadné zabudovat PHP kód rovnou do pohledů (View), tedy šablon s HTML kódem. Následně jsou všechny Blade šablony zkompilovány do PHP kódu a ukládány do mezipaměti. Příklad Blade šablon je možné vidět na následujících obrázcích (**3.3** a **3.4**). [8]

3.5.3 Homestead

Vývojáři Laravelu se snaží, aby tvorba nových projektů byla co nejsnažší. Proto jako součást Laravelu je vlastní vývojové prostředí Homestead, které nabízí jednoduchý a elegantní způsob správy a poskytování virtuálních strojů. Není tedy nutné na svůj lokální počítač instalovat PHP, webserver, ani jiný serverový software. Vše je v režii Laravelu. Toto prostředí je navíc multiplatformní a podporuje různé verze PHP a databázových serverů. Součástí prostředí je Composer, MySQL, zvolená verze PHP (závisí na verzi Laravelu), Apache, MariaDB, Git, Node, Redis a mnoho dalšího. [8]

Při vývoji byly potřebné informace o frameworku Laravel či souvisejících programových strukturách čerpány z oficiálních stránek vývojářů a na nich dostupné dokumentace. [8]

3.6 Composer

Composer [3] je velmi užitečný nástroj pro správu závislostí v PHP. Je multiplatformní a je nedílnou součástí mnoha PHP frameworků, jako například Laravel nebo Symfony. V angličtině bývá velmi výstižně označován jako "package manager", nebo také "dependency manager". U projektů vytvářených například ve frameworku Laravel (ne jinak je tomu i u této

```
<!-- Stored in resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('sidebar')
    @parent

    <p>This is appended to the master sidebar.</p>
@endsection

@section('content')
    <p>This is my body content.</p>
@endsection
```

Obrázek 3.3: Příklad Blade šablony pro sekce [8]

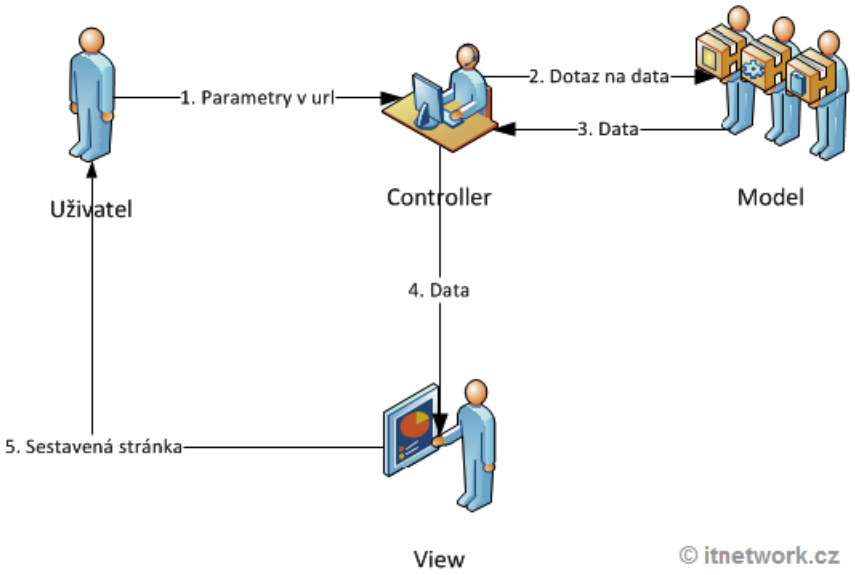
```
<!-- /resources/views/alert.blade.php -->

<div class="alert alert-danger">
    <div class="alert-title">{{ $title }}</div>

    {{ $slot }}
</div>
```

Obrázek 3.4: Příklad použití Blade šablony pro proměnné [8]

bakalářské práce) vznikají často závislosi na jiných již vytvořených balíčcích či knihovnách. Composer dokáže tyto balíčky a knihovny, na kterých je projekt závislý automaticky instalovat či později aktualizovat. Umožňuje rovněž automaticky načítat třídy daného projektu. Parametry pro Composer jsou uvedeny v souboru composer.json v kořenovém adresáři projektu. Zde jsou uvedeny veškeré závislosti, ať už na externích knihovnách, nebo například Laravel balíčcích, tak také na konkrétní verzi frameworku Laravel, nebo verzi jazyka PHP. [5]

3.7 Bootstrap

Bootstrap [2] je open source CSS framework vyvinutý vývojáři společnosti Twitter. Velmi usnadňuje práci se styly v nejrůznějších projektech využívajících HTML, JavaScript, ale hlavně CSS. Jde o jakousi knihovnu předem definovaných stylů, které je po importu toho frameworku možné používat. Bootstrap je také přímo součástí PHP frameworku Laravel,



Obrázek 3.5: Názorný diagram MVC architektury, jejíž rozšířením je framework Laravel [5]

který je použit pro tuto práci. Bootstrap je velice rozšířený a to hlavně díky své plné responzivitě, která umožňuje to, že styly jsou dokonale přizpůsobené jak pro klasické osobní počítače a notebooky, tak pro mobilní zařízení, jejichž uživatelé tvoří podstatnou část návštěvníků dnešních webových stránek. Pro Bootstrap existuje také mnoho volně využitelných šablon. [5]

3.8 Laravel Debugbar

Laravel Debugbar je balíček od vývojáře Barry Huevela, který je možné integrovat do Laravel projektu od verze 5. Balíček obsahuje ServiceProvider, který přidává debugbar do projektu a připojuje jej k výstupu. Uživatel při použití debugbaru má možnost vidět například, jaký uživatel je právě do systému přihlášen, jaká je odezva stránky, jaké byly použity šablony, nebo jaké dotazy do databáze proběhly a jak dlouho tvraly. Právě pro zmíněné funkce je tento užitečný nástroj použit i v této práci. [14]

Obrázek 3.6: Laravel Debugbar [14]

3.9 Verzovací systém Git

Git je distribuovaný systém pro správu verzí vytvořený Linusem Torvaldsem v roce 2005. Jedná se o open source projekt a v současné době existuje mnoho webových stránek poskytujících zdarma uložení zdrojových kódů vývojářů v repozitářích postavených na systému Git. Takovouto webovou stránkou s možností hostování zdrojových kódů je i stránka Bitbucket [1], která je využívána v této práci. Díky systému Git je možné mít dostupné své zdrojové kódy v mnoha verzích na jednom místě. Je možné se vracet zpět ke starším verzím, nebo jimi nahradit novou, například nepovedenou verzi. Dále je to také velmi užitečný nástroj pro práci v týmu, nicméně této vlastnosti nebylo pro tuto práci využito.

3.10 Jazyk UML

UML je zkratkou pro Unified Modeling Language a představuje jazyk pro grafické modelování, ať už funkčních částí systémů, databází či grafického rozhraní. Tento modelovací jazyk se tak rozšířil, že se stal standardem, a to hlavně díky tomu, že obsahuje všechny podstatné aspekty pro návrh a dá se v něm snadno orientovat. Bývá součástí mnoha materiálů, či dokumentací. Je to velmi užitečný nástroj pro návrh a následnou tvorbu například informačních systému. Pokud se někdo má vyznat v předloženém systému, je vždy pro pochopení jednodušší si projít dokumentaci včetně UML schémat, než složitě procházet zdrojový kód. Jazyk UML je velmi užitečný při komunikaci mezi programátory, vývojáři a projektanty, nebo zadavateli daného projektu.

UML diagramy nejčastěji vznikají při samotném návrhu informačního systému, kdy si vývojář nebo celý tým potřebují stanovit jasné cíle. Například pro určení toho, co daný informační systém má umět a jaké má mít funkce slouží často Use Case diagram, neboli diagram případu užití. Dále pro tyto účely mohou sloužit například detaily případu užití. Pro návrh relační databáze a jednotlivých tabulek se využívá ER (Entity Relationship) diagram. Vždy je pro pochopení daného problému či části informačního systému přijatelnější kreslená podoba nežli psaný text. Diagramy jsou také velmi důležité tím, že představují jistou úroveň abstrakce. Každý diagram může představovat pohled na systém z jiného úhlu. V danou chvíli daným diagramem je zachyceno pouze to, co je podstatné pro vývoj určité části systému a zbytek je zanedbán. [5]

Jazyk UML obsahuje mnoho diagramů a jejich různých variant. Pro účely této práce byl použit Use Case diagram a ER diagram.

3.10.1 Use Case diagram

Use Case Diagram, česky diagram případu užití, bývá prvním diagramem, který je vytvářen při tvorbě nového informačního systému. Tento diagram popisuje chování systému z pohledu různých typů uživatelů. Zobrazuje tedy pohled uživatele na systém. Záměrem tohoto diagramu je popsat, jak systém funguje jako celek, avšak není zde popsáno, jak je tato funkčnost realizována. Je zde tedy použita určitá míra abstrakce. Díky těmto vlastnostem je Use Case diagram velmi významnou součástí návrhu systému, zejména pak v počátcích, kdy je důležité stanovit si, co bude daný informační systém umět.

Diagram se skládá z aktérů (actor) a případů užití (use case). Případ užití je jakousi černou skříňkou (black box), jelikož představuje určitou funkcionalitu, nebo dosažení nějakého cíle, avšak zůstává skryto, jak je toho dosaženo. Například konkrétně v této práci je to přidání příspěvku. V diagramu toto bude zapsáno pouze jako přidání přísvěvku, ale

ve skutečnosti se jedná o vyplnění formuláře pro přidání příspěvku a zapsání do databáze. Aktér je naopak role nějakého uživatele, jako třeba běžný uživatel, vedoucí týmu, řešitel projektu a podobně. Může to být také čas či jiný externí systém. [5]

3.10.2 ER diagram

ER diagram, nebo také Entity Relationship diagram, bývá spolu s Use Case diagramem jedním z hlavních diagramů, které vznikají při návrhu nového informačního systému. ER diagram je základním diagramem pro návrh databáze. Udává jednotlivé entity či entitní množiny a jejich atributy. Tyto entity jsou později při tvorbě samotné databáze realizovány jako databázové tabulky a jejich atributy jako jednotlivé sloupce každé tabulky. Již v tomto diagramu se určují primární a cizí klíče pro každé entity (tabulky). Existují zde tedy i pojmenované vztahy mezi entitami a kardinality těchto vztahů. ER diagram velmi výrazně usnadňuje pozdější tvorbu databáze. [5]

Kapitola 4

Analýza a specifikace požadavků

Velmi důležitou a také jednou z klíčových částí při vývoji informačního systému je analýza a specifikace požadavků. Bez zvládnuté analýzy požadavků se může stát, že vývojář vytvoří velmi dobrý a fungující informační systém, avšak neodpovídající požadavkům zákazníka. Takový systém je tedy k ničemu. Zde je nutné dobře zanalyzovat skutečné požadavky zákazníka nebo zadavatele projektu na systém. Až poté může začít vznikat návrh nového systému.

Po domluvě s vedoucím této bakalářské práce panem Ing. Vladimírem Bartíkem, Ph.D. byl vytvořen a konzultován seznam požadavků, které má nově vytvořený informační systém pro řízení projektů v IT firmě splňovat. Tyto požadavky byly schváleny vedoucím práce a budou detailněji popsány v několika následujících odstavcích.

Na úvod je nutné uvést, že v daném systému existuje několik rolí uživatelů, které jsou:

- řadový uživatel,
- člen týmu,
- řešitel projektu,
- vedoucí týmu,
- vedoucí projektu a
- administrátor.

Uživatelé mají oprávnění vykonávat různé akce právě v zásvislosti na jejich rolích.

4.1 Přidání nových uživatelů, přihlášení, úprava profilu a dashboard

Nové uživatele může do systému přidávat pouze administrátor informačního systému. Ten vytvoří účet novému uživateli, přičemž zadá jeho jméno, příjmení, emailovou adresu a heslo. Na uvednou emailovou adresu je pak doručena uživateli zpráva s jeho přihlašovacími údaji. Uživatel má pak možnost se z přijatého emailu přímo "prokliknout"přes odkaz do systému a přihlásit se.

Každý uživatel se tedy musí před vstupem do systému přihlásit. Toto opatření zamezí vstupu do systému neoprávněným osobám. Po přihlášení je uživatel přesměrován na stránku

s dashboardem. Přihlášený uživatel se může odhlásit, nebo také upravit údaje ve svém profilu, či změnit heslo ke svému účtu. Toto smí vykonávat také administrátor sytému a to nad všemi uživatelskými účty. Dále si může přihlášený uživatel zobrazit přehled týmů, projektů a ostatní registrované uživatele.

4.2 Týmy

Každý uživatel má možnost vytvořit tým a stává se automaticky jeho vedoucím. Vedoucí může přidávat nové členy do týmu a také z týmu odebírat stávající členy. Dále upravuje informace o týmu, jako například jeho zkratku, celý název nebo popis. U týmů je možné nastavit také oprávnění. To určuje, kdo může vidět týmové příspěvky, zda jen členové, nebo všichni uživatelé systému. Pokud je nastaveno oprávnění zobrazovat příspěvky pouze pro členy týmu, pak ostatní uživatelé vidí pouze informace o týmu, kdo je jeho vedoucí a kdo jsou členové. Toto nastavení opět spravuje vedoucí týmu. Vedoucí týmu má také možnost celý tým rozpustit.

4.2.1 Týmové příspěvky

Každý člen týmu má možnost přidávat týmové příspěvky. Jedná se o příspěvky ve stylu blogu. Tato možnost slouží jako komunikační kanál pro členy týmu. Ti zde mohou vyvěšovat různé upomínky, rozpis a harmonogram dalších jednání s klientem, plán na další týden a podobně. Na stránce týmu se zobrazují náhledy dvou nejnovějších příspěvků. Uživatelé si také mohou zobrazit přehled všech příspěvků daného týmu a u každého příspěvku pak jeho detail. Příspěvky jsou typu info, požadavek a upozornění a ke každému je možné také přidat obrázek. Pro to, aby u příspěvků mohla probíhat diskuze, existuje zde možnost přidávat komentáře pod každý příspěvek. Tímto mohou například ostatní členové týmu vyjádřit své připomínky k časovému harmonogramu a podobně.

4.3 Projekty

Stejně jako může každý uživatel zakládat týmy, může také vytvořit nový projekt. Opět se automaticky stává jeho vedoucím. Může přidávat a odebírat řešitele a upravovat informace týkající se daného projektu, jako je zkratka, celý název nebo popis. Vedoucí také nastavuje oprávnění. Podobně jako u týmu jde o to, kdo může vidět různé detaily projektu. V tomto případě se jedná o úkoly. Vedoucí projektu tedy určuje zda úkoly u daného projektu mohou vidět pouze jeho členové nebo i ostatní uživatelé. Učiní tak pomocí volby oprávnění s možnostmi členové, nebo všichni.

Vedoucí projektu má možnost také uzavřít projekt. Pokud tuto možnost zvolí, pak status takového projektu se změní z aktivní na uzavřen (nově vytvořený projekt má implicitní status aktivní). Pokud je projekt uzavřen, pak není možné přidávat žádné nové řešitele, či odebírat stávající. V takovém případě není možné ani přidávat nové úkoly k projektu, či přidávat nebo odebírat jakékoliv soubory. Vedoucí má však možnost znovu otevřít projekt. Jeho status se tedy opět změní na aktivní a výše zmíněné funkce jsou znovu dostupné, jako například ihned po vytvoření projektu. Vedoucí má také možnost celý projekt odstranit, včetně všech souborů a úkolů s ním souvisejících.

4.3.1 Úkoly

Řešitelé projektů mohou přidávat různé úkoly, jako například vytvoření nového modulu k internetovému obchodu, úprava databáze, doplnění některých částí projektu, oprava chyby a další. Tyto úkoly pak mohou být přiřazovány jednotlivým řešitelům projektu. Úkol může řešit jeden či více uživatelů. Při vytvoření, či následné úpravě úkolu je určen jeho typ. Úkol může být typu požadavek, jako například doplnění funkčnosti určité části projektu, nebo chyba. U každého úkolu je také uveden jeho stav, který nabývá hodnot nový, řeší se a dokončen.

Pod každý úkol mohou být přidány komentáře, kde uživatelé uvádí své připomínky nebo nápady k danému problému.

4.3.2 Soubory

Ke každému projektu, jehož status je aktivní, je možné nahrávat soubory ze zařízení (počítače). Každý soubor je klasifikován, buď jako soubor se zadáním k danému projektu, nebo jako soubor k odevzdání. Veškeré nahrané soubory mohou být stahovány všemi členy týmu.

Soubory je možné také přidávat k úkolům. Zde platí to samé jako u souborů přidaných k projektu. U souborů přidaných k úkolu je navíc možnost kopírovat nebo přesunout soubor z úkolu do projektu, ke kterému daný úkol náleží.

4.4 Více jazyků

Uživatelé přihlášení do systému mají možnost změnit jazyk systému dle jejich potřeb či preferencí. V nabídce jsou tři základní jazyky a to čeština, angličtina a slovenština.

4.5 Notifikace

Po některých akcích provedených přihlášeným uživatelem jsou zaslány notifikace jiným uživatelům systému. Jedná se o notifikace v podobě elektronické pošty zaslané na emailovou adresu uživatelům, kterým jsou určeny. Pokud je například odstraněn projekt ze systému, jsou všichni řešitelé tohoto projektu obeznámeni o této skutečnosti. Dalším případem je, když administrátor přidá nového uživatele do systému. Tomu jsou následně zaslány přihlašovací údaje. Všechny notifikace je možné vidět v Use Case diagramech v následující kapitole Návrh systému.

4.6 Rychlost získávání dat

Vždy, když je potřeba zobrazit například seznam všech registrovaných uživatelů, seznam projektů, detail nějakého týmu, nebo provádět další akce, jako třeba přidání příspěvku či komentáře, tak se informační systém dotáže příslušným způsobem do databáze, kde jsou tyto informace uložené a vhodným způsobem je prezentuje uživateli.

Projekt je zaměřen na rychlost získávání dat. Jde o snahu co nejvíce zkrátit dobu, kterou trvá získání dat z databáze a jejich zpracování. Laravel pro práci s databází používá své nástroje Query Builder a Eloquent. V této práci je právě nástroj Eloquent nahrazen vlastním řešením (viz dále).

4.7 Use Case diagramy

Výstupem analýzy a specifikace požadavků bývá nejen soupis požadavků na systém, ale také Use Case diagram, který tyto požadavky zobrazuje přehledně ve formě jednoho, či více diagramů. Pro vývojáře či programátora je mnohem jednodušší převést požadavky do zjednodušeného diagramu a podle něj postupovat při tvorbě systému. Po pochopení požadavků na systém bývá tedy vytvořeno několik diagramů reprezentujících vytvářený systém. Tyto diagramy obsahují jistou míru abstrakce požadavků. Samotná implementace daných částí či požadavků na funkčnost bývá v návrhu skryta.

Use Case diagram byl rozdělen do několika samostatných částí dle aktérů a jejich případů užití. V systému existuje několik typů uživatelů a tedy i aktérů v diagramu. Aktéři jsou označeni písmenem A (Actor), pořadovým číslem a názvem aktéra. Případy užití pak písmeny UC (Use Case), pořadovým číslem a stručným označením akce, kterou reprezentují.

V následujících odstavcích budou stručně popsány jednotlivé části Use Case diagramu, ke kterým bude vždy přiložen odpovídající diagram. Následuje krátké představení a stručný popis ER diagramu.

4.7.1 Diagram pro aktéry Nepřihlášený uživatel a Přihlášený uživatel

Na následujícím diagramu (obrázek **5.1**) se náchází aktéři Přihlášený uživatel a Nepřihlášený uživatel. Nepřihlášený uživatel může provést pouze akci, která jej přihlásí do systému. Přihlášený uživatel může spravovat svůj profil, což označuje změnu osobních údajů a hesla, změnit jazyk sytému (čeština, angličtina, slovenština), změnit databázový driver a zobrazit profily ostaních uživatelů. Dále může také zobrazit týmy, založit vlastní tým, zobrazit projekty a založit vlastní projekt.

4.7.2 Diagram pro aktéra Člen týmu

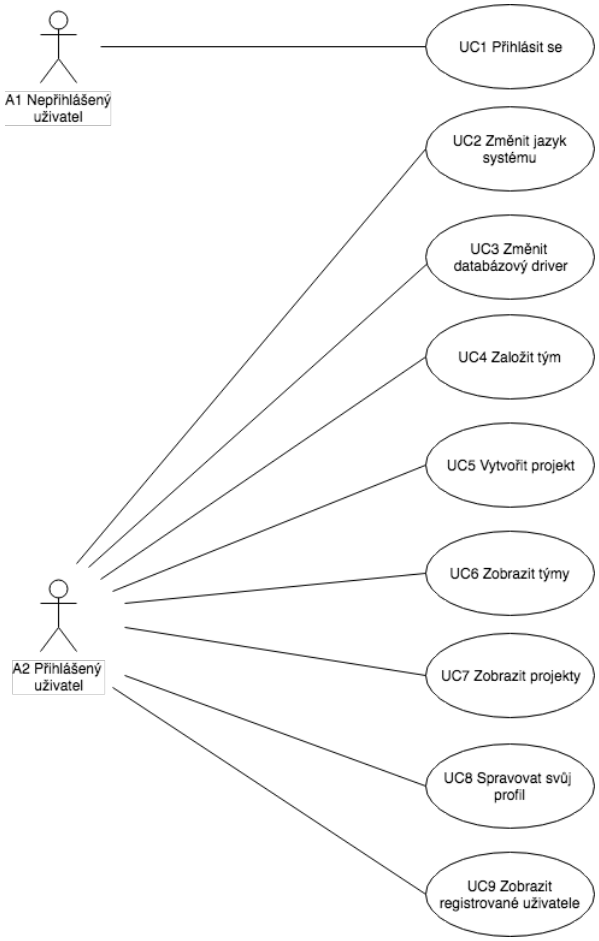
Jak je vyobrazeno na obrázku **5.2**, tak aktér Člen týmu může provádět stejné akce jako aktér Přihlášený uživatel (viz šipka). Jelikož je členem nějakého týmu, pak může tento tým opustit. Dále může přidávat týmové příspěvky, které představují články ve stylu blogu. Přidáním příspěvku jsou zaslána oznámení všem členům týmu. Aktér Člen týmu může také přidávat komentáře k příspěvkům, přičemž je zasláno oznámení o této akci autorovi příspěvku, upravit a odstranit vlastní příspěvky, nebo upravit a odstranit vlastní komentář.

4.7.3 Diagram pro aktéra Řešitel projektu

Na obrázku **5.3** je možné vidět, že aktér Řešitel projektu může provádět velkou spoustu akcí, včetně těch, které může provádět Přihlášený uživatel. Mezi nejpodstatnější patří přidávání, úprava a odebírání úkolů týkajících se daného projektu, nahrávání a stahování souborů a manipulace s nimi. Samozřejmě tento aktér může také opustit daný projekt. Nutno dodat, že po opuštění projektu není možné se opět vrátit. Přidávat řešitele může pouze vedoucí příslušného projektu.

4.7.4 Diagram pro aktéra Vedoucí týmu

Diagram pro aktéra Vedoucí týmu je zobrazen na obrázku **5.4**. Aktér Vedoucí týmu je ten, který založí nějaký tým a stává se tak automaticky jeho vedoucím. Z diagramu lze vyčíst, že má možnost vykonávat stejné akce jako člen týmu a navíc několik dalších. Mezi ně



Obrázek 4.1: Diagram případů užití pro aktéry Nepřihlášený uživatel a Přihlášený uživatel

patří úprava nebo rozpuštení týmu, přidávání a odebírání členů týmu, a také odstraňování příspěvků týmu a jejich komentářů. Vedoucí týmu by mohl být označen také jako správce týmu.

4.7.5 Diagram pro aktéra Vedoucí projektu

Názorný diagram obsahující aktéra Vedoucí projektu včetně příslušných případů užití je na obrázku **5.5**. Vedoucí projektu rozšiřuje aktéra Řešitel projektu o několik dalších akcí. Má možnost upravit projekt, nebo jej celý odstranit, odebírat soubory náležící projektu, nebo s ním souvisejícím úkolům. Dále může přidávat nebo odebírat řešitele daného projektu, a upravit nebo odstranit úkoly. Při některých akcích proběhne také notifikace uživatelů, kterých se tato akce týká, jako například při přiřazení uživatele jako řešitele projektu, nebo odstranění projektu.

4.7.6 Diagram pro aktéra Administrátor systému

Aktér Administrátor systému je v podstatě správcem celého systému. Může vykonávat akce jako aktéři Vedoucí týmu a Vedoucí projektu a k tomu ještě několik dalších. Ty jsou uvedeny na diagramu na obrázku **5.6** a patří mezi ně například přidání nového uživatele

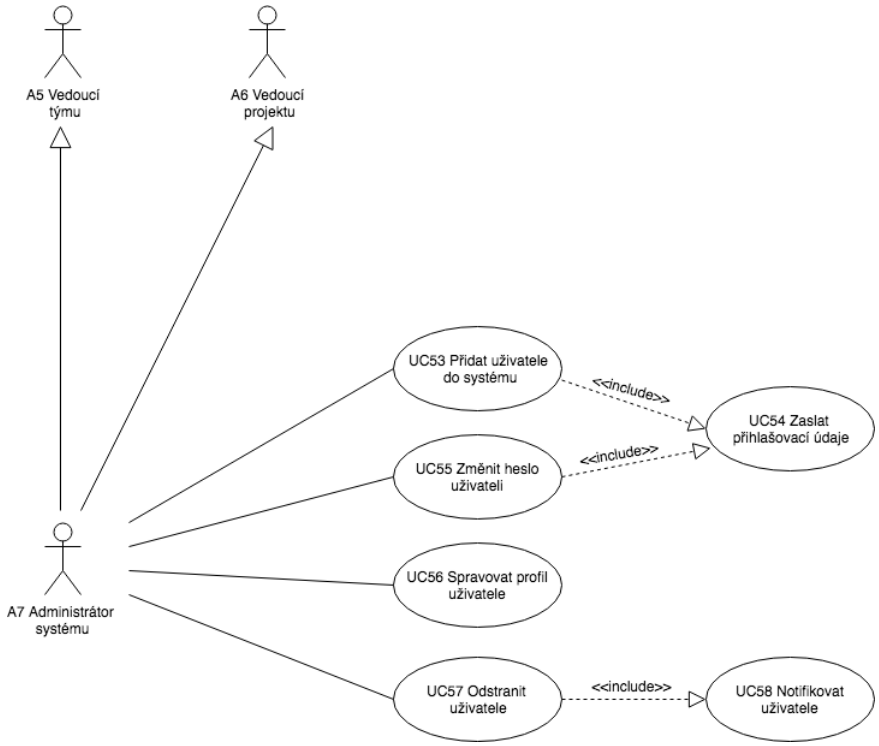
Obrázek 4.2: Diagram případů užití pro aktéra Člen týmu

do informačního systému, odebrání stávajícího uživatele, nebo správa profilů jednotlivých uživatelů a změna jejich hesel. Při přidání uživatele do systému, nebo při změně hesla některému uživateli je odeslán email s přihlašovacími údaji do systému. Při odstranění uživatele ze systému je mu tato skutečnost oznámena také pomocí elektronické pošty na příslušnou adresu.

Obrázek 4.3: Diagram případů užití pro aktéra Řešitel projektu

Obrázek 4.4: Diagram případů užití pro aktéra Vedoucí týmu

Obrázek 4.5: Diagram případů užití pro aktéra Vedoucí projektu



Obrázek 4.6: Diagram případů užití pro aktéra Administrátor systému

Kapitola 5

Návrh systému

Stejně jako analýza a specifikace požadavků hraje návrh systému velmi významnou roli při jeho tvorbě. Analýzou a specifikací požadavků zjistí vývojář co se od nového systému očekává a jaká základní specifika má splňovat. Jsou zde upřesněny některé detaily, které jsou podrobněji rozvedeny pro lepší pochopení oběma stranami. Výstupem analýzy a specifikace bývá většinou psaný text rozdělený do několika odrážek či odstavců pro jednotlivé podúkoly. Takový text by měl zahrnovat všechna důležitá specifika systému a nemělo by zde být nic opomenuto. Součástí výstupu analýzy a specifikace požadavků bývá také Use Case diagram, ve kterém se může vývojář jednoduše orientovat a stanovit tak snadno dílčí cíle při vývoji systému.

Při tvorbě této bakalářské práce, tedy při návrhu a tvorbě informačního systému pro řízení projektů v IT firmě byly s panem Ing. Vladimírem Bartíkem, Ph.D. nejprve specifikovány požadavky na systém a poté byly vytvořeny Use Case a ER diagramy. Jak je uvedeno výše, tyto diagramy jsou základními prostředky pro návrh informačních systémů a jsou součástí jazyka UML. V této kapitole byl vytvořen ER diagram pro návrh relační databáze.

5.1 ER diagram

Nedílnou součástí informačních systému je databáze, ve které jsou uchována veškerá perzistentní data. Tato data představují obsah informačního systému a ne jinak je tomu i u této bakalářské práce. Důležité pro chod a fungování programu je vhodně navržená databáze. Při špatně navržené databázi se mnohdy problémy objevují až v průběhu vývoje samotného systému. Poté je velmi obtížné tyto problémy a chyby odstraňovat. Pro návrh databáze se využívá ER diagram.

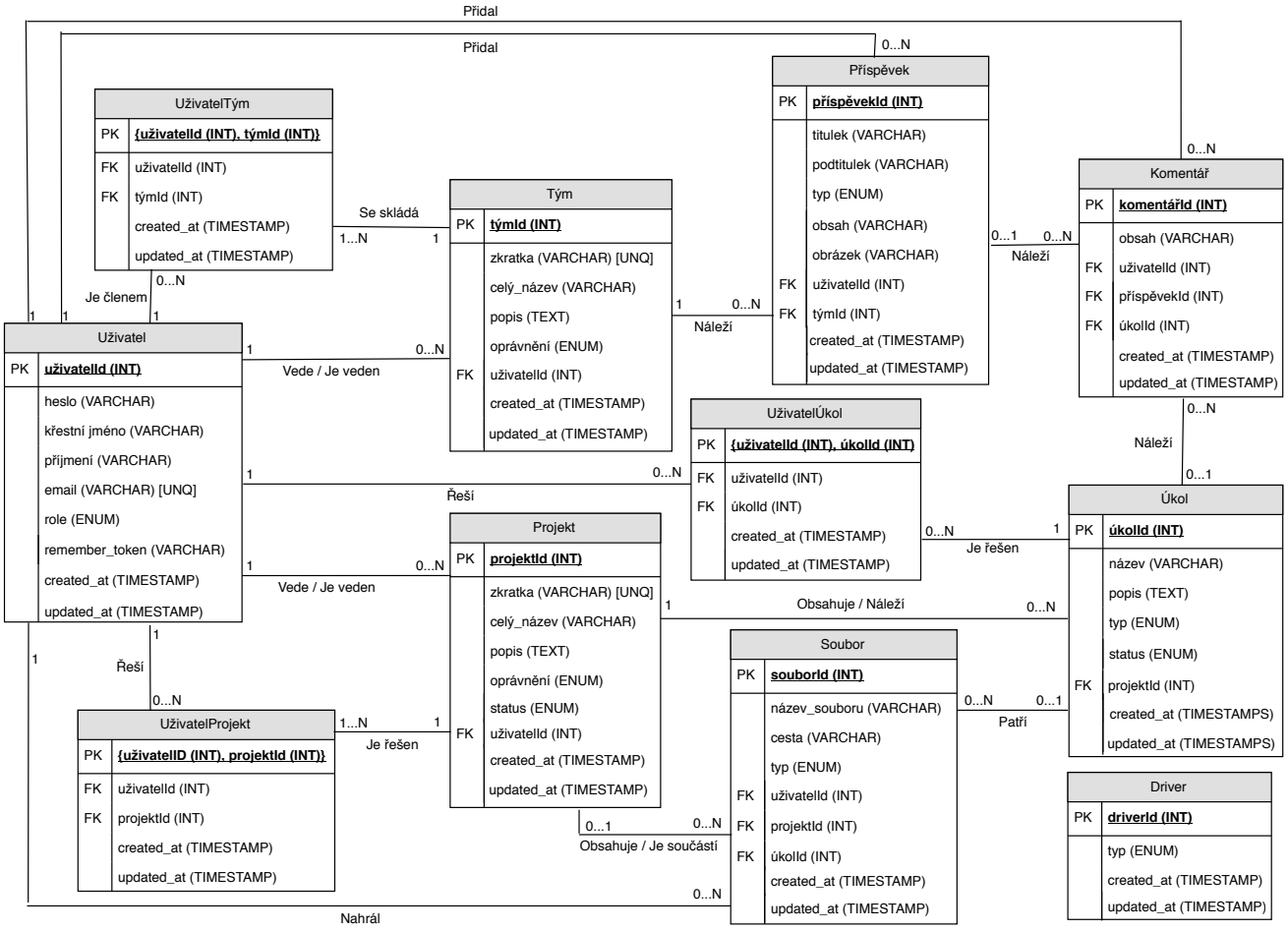
Jak je možné vidět na obrázku **5.7**, tak ER diagram pro databázi k informačnímu systému pro řízení projektů v IT firmě se skládá z několika entitních množin. Každá z nich obsahuje atributy různých datových typů, ať už varchar, text, enum (výčet), int nebo timestamps (časová značka - zpravidla datum a čas). Zkratka UNQ znamená, že daná hodnota musí být vždy v rámci dané entitní množiny unikátní. Jako příklad může posloužit atribut email v entitní množině Uživatel. Pokud entitní množina obsahuje entitu, jejíž hodnota atributu email je například jmeno.prijmeni@gmail.com, pak se v dané entitní množině nesmí vyskytovat žádná jiná entita, jejíž hodnota atributu email je totožná. Dalšími zkratkami jsou FK a PK. PK znamená primary key, v překladu primární klíč, který slouží jako jednoznačný identifikátor entity v rámci dané entitní množiny a jeho hodnota je tedy stejně jako

hodnota atributu označeného UNQ unikátní v rámci dané entitní množiny. FK znamená foregin key, tedy cizí klíč. Hodnota cizího klíče se musí shodovat s hodnotou primárního klíče některé jiné entity.

Entitní množiny představují reálné prvky informačního systému jejichž data je nutné uchovat v databázi. Množina Uživatel reprezentuje zaregistrované uživatele systému, o kterých jsou uchovány informace, jako křestní jméno, příjmení, email a další. Entitní množina Projekt reprezentuje všechny vytvořené projekty v systému, stejně jako Tým představuje všechny jednotlivé týmy do nichž jsou uživatelé zhlukováni. O těchto objektech se uchovávají informace jako zkratka, celý název, nebo kdo je založil, popřípadě vytvořil. Entitní množina Příspěvek reprezentuje týmové příspěvky a Komentář představuje komentáře k těmto příspěvkům, ale také k úkolům, které reprezentuje entitní množina Úkol. Zajímavou entitní množinou může být množina s názvem Driver. V databázi se uchová údaj o tom, jaký databázový driver se aktuálně používá, zda implicitní s využitím Eloquentů, nebo alternativní, který nahrazuje Eloquenty vlastními objekty.

Mezi entitními množinami existují určité vztahy, které velmi úzce souvisí s cizími klíči (FK). Například cizí klíč uživatelId v entitní množině Komentář musí obsahovat stejnou hodnotu, jakou má atribut uživatelId v některé entitě entitní množiny Uživatel. Toto určuje, že každý komentář musel napsat nějaký uživatel.

Vazební entitní množiny jako například UživatelProjekt řeší M:N problém, kde jeden uživatel může být řešitelem více projektů a naopak jeden projekt může řešit více uživatelů.



Obrázek 5.1: ER diagram

Kapitola 6

Implementace

V této kapitole jsou popsány jednotlivé části samotné implementace informačního systému pro řízení projektů v IT firmě. Celá implementace vychází z analýzy a specifikace požadavků a následného návrhu systému. Při tvorbě funkčnosti systému byly hlavní předlohou Use Case diagramy z předchozí kapitoly (kapitola 5 - Návrh systému), doplněné o podrobnější specifikaci požadavků z kapitoly 4 - Analýza a specifikace požadavků. Databáze byla vytvořena podle ER diagramu popsaného také v předchozí kapitole. Pro implementaci byly využity prostředky a technologie, jako například framework Laravel, PHP, MySQL, HTML a další popsané v kapitole 3 - Použité technologie.

6.1 Laravel balíček

Hlavním způsobem jak přidávat další funkce do aplikace založené na frameworku Laravel jsou balíčky. Balíčků je plný Github a další úložné prostory na internetu. Některé jsou volně dostupné, některé naopak za poplatek. Vývojáři pracující s frameworkem Laravel většinou, když vytvoří nový projekt, nebo rozšíření nějaké funkčknosti Laravelu, tak tak učiní pomocí Laravel balíčku. Pro zavedení balíčku stačí přidat jeho referenci do Composeru v již vytvořeném projektu a splnit několik dalších instalačních kroků uvedených obvykle v README souboru. Pokud je samotný balíček rovnou celý projekt, jako v případě této bakalářskě práce, kdy balíček představuje celý informační systém pro řízení projektů v IT firmě, pak je vytvořen nový projekt Laravel a do něj balíček vložen. Pokud by měl být systém za nějakou dobu upraven, nebo rozšířen, pak se tak učiní pomocí dalšího balíčku. Tímto je zaručena snadná rozšiřitelnost, spravovatelnost a upravitelnost systému. Díky těmto vlastnostem je systém realizovaný v rámci této práce zhotoven jako samostatný balíček.

6.2 Composer

Composer je nástroj pro správu závislostí v PHP. Jedná se o jakýsi "package manager". Jak vyplývá z předchozích dvou vět, composer je užitečný nástroj při tvorbě Laravel aplikací a zejména pak balíčků. Umožňuje přidávat funkčnost do projektu pomocí balíčků, nebo například zajišťuje, že daná aplikace může být použita na zařízení (například serveru) s určitými zdroji, jako jsou například PHP v uvedené a vyšší verzi a Laravel framework v určité verzi. Dále v souboru určeném pro zpracování Composerem (soubor composer.json) bývají uvedeny informace o autorovi projektu a také o projektu samotném. Pokud jsou v balíčku nebo aplikaci provedeny nějaké změny (přidány, nebo odebrány soubory, či je změnen ně-

který jmenný prostor), pak stačí pouze spustit příkaz composer update v příkazové řádce v kořenovém adresáři projektu a Composer si načte automaticky nové závislosti. Soubor pro zpracování Composerem vytvořený v rámci této bakalářské práce je na obrázku **6.1**.

```
{
    "name": "xkamen06/pms",
    "description": "Project Management System (web information system)",
    "type": "project",
    "authors": [
        {
            "name": "Zdenek Kamensky",
            "email": "xkamen06@stud.fit.vutbr.cz"
        }
    ],
    "minimum-stability": "stable",
    "require": {
        "php": ">=7.1",
        "laravel/framework": "5.5.*",
        "predis/predis": "^1.1",
        "roave/security-advisories": "dev-master"
    }
}
```

Obrázek 6.1: Soubor composer.json vytvořený pro balíček s informačním systémem pro řízení projektů v IT firmě

6.3 Service Provider

Dalším z hlavních prvků frameworku Laravel a jeho balíčků jsou Service Providery. Nově vytvořené aplikace, stejně jako všechny hlavní služby společnosti Laravel, jsou zavedeny právě pomocí Service Providerů. Jedná se o třídy, které obsahují metody pro registraci služeb, middleware, rout, migrací, pohledů (views) a dalších prvků. Jinými slovy je zde obsažena konfigurace každého balíčku aplikace. Proto každý balíček obsahuje (minimálně jeden) Service Provider, který se zpravidla umísťuje do kořenového adresáře daného balíčku. Každý používaný Service Provider musí být sám také zaregistrovaný, a to v poli providers souboru app.php v adresáři config konkrétní instalace projektu Laravel. Bez tohoto kroku není na Service Provider daného balíčku brán zřetel a zamezuje to tak jeho využívání.

V této práci byl vytvořen PMSServiceProvider, který ve své hlavní metodě (boot) registruje všechny migrace balíčku, routy a pohledy. Dále tato metoda volá metodu pro zveřejňění(publishing) například jazykových souborů pro překlad systému, konfiguračních souborů, pohledů pro přihlašování do systému a dalších. Poslední, avšak velmi důležitou funkcí tohoto Service provideru je registrace singletonů. Tato funkce stručně řečeno přiřazuje rozhraní (interface) repozitářů k jednotilivým repozitářům podle aktuálního databázové driveru. Vychází přitom z konfiguračního souboru singletons.php, ve kterém jsou jednotlivé cesty k rozhraním a repozitářům uchovány. To jaký se právě používá databázový driver je uloženo v databázi, popřípadě v mezipaměti (cache).

6.4 Tvorba databáze

Základním stavebním kamenem informačního systému je databáze. Proto hned v začátku vývoje informačního systému je nutné podle sestaveného ER diagramu vytvořit funkční databázi a napojit ji na nově vznikající systém. Pro vývoj a následnou prezentaci projektu nejsou potřeba žádná reálná data uložená v databázi, nýbrž postačují data určená pro demonstraci chodu systému. Tato data však musí mít určitou strukturu, aby alespoň připomínala reálná data, která budou po nasazení do databáze vkládána. Pokud se demonstrační data co nejblíže svoji strukturou blíží datům reálným, předejde se tak možné nekonzistenci databáze po nasazení sytému do produkčního prostředí. Pro vytvoření databázových tabulek včetně jednotlivých sloupců a určení všech cizích klíčů a dalších záležitostí se používají ve frameworku Laravel takzvané migrace. Pro vytvoření a vložení demonstračních dat do databáze se využívají seedy.

6.4.1 Migrace

Migrace slouží k vytvoření a úpravě databázových tabulek. Pomocí migrací je možné snadno a rychle vytvořit databázovou tabulku, nebo například do některé tabulky přidat jeden sloupec, nebo také nějaký sloupec odebrat a podobně. Migrace se vždy vytváří jako jednoduché třídy, které rozšiřují Laravelovskou třídu Migration. Název třídy se volí vhodně a výstižně vzhledem k danému úkonu, který má být vykonán. Pro vytvoření tabulky team je třída pojmenenována například CreateTeamTable. Aby nedocházelo k tomu, že je vytvořena nějaká databázová tabulka, která obsahuje cizí klíč, který je primárním klíčem ještě nevytvořené tabulky, jsou migrace spouštěny chronologicky, tak jak byly vytvořeny. To je zaručeno tím, že názvy souborů všech migrací začínají datem jejího vytvoření. Každá migrace obsahuje dvě základní metody, up a down. Do metody up se většinou umisťuje vytvoření nějaké tabulky, nebo její úprava a do metody down potom zrušení tabulky nebo úpravy. Věškeré tyto výše zmíněné úkony jsou realizovány pomocí fasády Schema. Ovládání migrací se provádí pomocí několika příkazů v příkazové řádce v kořenovém adresáři projektu. Seznam příkazů, podrobnější popis migrací a popis fasády Schema je možné najít na oficiálních stránkách Laravelu [8] a k nim přiložené dokumentaci.

6.4.2 Seedy

Seedy slouží jako nástroj pro vkládání demonstračních dat do databáze. Jedná se o jednoduché třídy. Jak bylo napsáno výše, pro vývoj a prezentaci nového systému nejsou potřeba často obtížně získaná reálná data, ale postačí data demonstrační. Nutné však je, aby tato data se svojí strukturou blížila reálným datům. Seedy obsahují metodu run, která je spuštěna při běhu daného Seedu. V této metodě obvykle bývá cyklus, který vkládá data do databáze. Někdy není reálné, aby si vývojář systému všechna data z hlavy vymýšlel (například jména a příjmení stovek osob). Pro tyto účely byl v této bakalářské práci použit balíček Faker [20] od vývojáře Francoise Zaninotta. Jedná se o PHP knihovnu, která umožňuje generování nereálných demonstračních dat. Faker je používán pomocí fasády Faker. Tímto způsobem jsou vložena demonstrační data do databáze systému realizovaného v rámci této práce.

6.5 Routy

Po vytvoření a naplnění databáze demonstračními daty je nutné vytvořit v informačním systému realizovaném ve frameworku Laravel routy. Ty slouží k přesměrování z URL na odpovídající metodu controlleru (řadiče). Seznam rout tedy obsahuje seznam různých URL a k nim přiřazené metody controllerů, které budou volány. Routy mohou být pojmenované, takže například po kliknutí na tlačítko se nemusí přecházet na URL adresu, ale odkazování probíhá podle jména routy. Routy rozlišují několik HTTP metod. V této práci jsou využívány metody GET a POST. Pomocí URL mohou být předány parametry, jako například id zobrazeného uživatele. Takto je to řešeno při použití HTTP metody GET. Při použití POST jsou parametry předány v request objektu.

V této práci jsou routy rozděleny pro přehlednost do několika souborů, z nichž každý je zaměřen na jinou část sytému. Na následujícím obrázku (**6.3**) je příklad rout týkajících se uživatelů informačního systému.

```
Route::get('/allusers', 'UserController@showAllUsers')->name('allusers');
Route::get('/userprofile/{userId}', 'UserController@showUserProfile')->name('userprofile');
Route::get('/edituser/{userId}', 'UserController@showEditForm')->name('edituser');
Route::post('/updateuser/{userId}', 'UserController@updateUser')->name('updateuser');
```

Obrázek 6.2: Příklad konkrétních rout z informačního systému pro řízení projektů v IT firmě

6.6 Controllery

Controllery, neboli řadiče jsou hlavním prvkem frameworku Laravel, který je založený na MVC (Model View Controller) architektuře. Zde se odehrává hlavní část logiky informačního systému. V controllerech tedy bývá centralizována logika zpracování požadavků. Samotný controller představuje třídu obsahující metody pro různé typy požadavků. Controllerů je několik, přičemž každý obsluhuje požadavky jiné části systému. V této práci jsou vytvořeny následující:

- ArticleController,
- CommentController,
- DashboardController,
- DiverController,
- FileController,
- ProjectController,
- TaskController,
- TeamController a
- UserController.

Z předchozího výčtu je patrné, že každý controller obstarává samostatnou logickou část systému a další souvislosti s ní spjaté. Controllery obsahují metody, které jsou volány pokud zadaná URL koresponduje s příslušnou routou (popřípadě je na routu odkazováno pomocí jejího názvu). V opačném případě proběhne vyjímka NotFoundException. Do metody jsou pomocí routy často poslány parametry, jako například id týmu, id článku a podobně. Příslušná metoda pomocí repozitářů získá potřebná data z databáze, nebo jiného média. V případě této práce jsou data získána vždy z databáze, nebo mezipaměti. Poté je vrácen odpovídající pohled (view), do kterého jsou vložena získaná data a ten je vhodným způsobem prezentován uživateli systému. Pomocí jiných metod jsou například upraveny záznamy v databázi, nebo jsou rovnou odstraněny. Prostřednictvím dalších jsou do databáze vkládány nové záznamy. Tyto akce neprovádí jen metody controllerů, ale činí tak pomocí repozitářů. Na uvedeném principu fungují všechny controllery a jejich metody v tomto systému. Pokud požadovaná data v databázi neexistují je opět provedena vyjímka NotFoundException. To však realizují repozitáře.

V rámci tohoto projektu mají controllery další významnou funkci. Zajišťují práva uživatelů vykonávat různé akce. Každý controller obsahuje jednu, nebo více metod pro autorizaci uživatele. Pokud například některý uživatel chce upravit projekt, pak je nejprve volána metoda pro autorizaci uživatele k úpravě projektu s id příslušného projektu a v této metodě je vyhodnoceno, zda na to má uživatel právo. Pokud ano, je akce provedena. Pokud ne, je opět provedena vyjímka, nicméně tentokrát typu UnauthorizedException s chybovou hláškou "Permission denied.". Příklad autorizační metody je na obrázku (**6.4**). Tato metoda určuje, zda je uživatel oprávněn přidat nový úkol k projektu. Jak je možné z metody vyčíst, přidat nový úkol může administrátor systému, vedoucí projektu nebo člen projektu.

```php
/** Authorize user, if user is able to do the action ...*/
public function authorizeUserCreate(int $projectId) : void
{
    $project = projectRepository()->getProjectById($projectId);
    if (auth()->user()->role !== 'admin' && auth()->user()->id !== $project->getLeaderId()
        && !$project->isMember(auth()->user()->id)) {
        throw new UnauthorizedException('Permission denied.');
    }
}
```

Obrázek 6.3: Autorizační metoda pro vytvoření nového úkolu k projektu

Controllery také uchovávájí informaci o tom, kolik záznamů se má zobrazit na jednu stránku, pokud má být vypsán například seznam všech projektů, nebo týmů.

Na obrázku **6.5** je příklad typické metody controlleru, která slouží pro zobrazení všech příspěvků týmu, který je určen prostřednictvím id uloženého v parametru metody. V metodě je nejprve pomocí repozitáře pro týmy získán objekt daného týmu, následně je zavolána metoda pro autorizaci (zda přihlášený uživatel má právo zobrazit příspěvky), poté jsou získány objekty všech příspěvků daného týmu pomocí repozitáře pro příspěvky a na závěr je vrácen příslušný pohled, do kterého jsou předána získaná data. Pohled pak prezentuje data vhodnou formou uživateli.

```php
/** Shows all articles by team (identified by teamId) ...*/
public function showArticlesByTeamId(int $teamId) : View
{
    $team = teamRepository()->getTeamById($teamId);
    $this->authorizeUserShow($team);
    $articles = articleRepository()->getAllArticlesByTeamId($teamId);
    return view('pms::Article.indexbyteam', compact('articles', 'team'));
}
```

Obrázek 6.4: Metoda pro zobrazení všech příspěvků týmu určeného prostřednictvím id

6.7 Pohledy (Views)

Pohledy ve frameworku Laravel, stejně jako v čisté architektuře MVC, slouží k vhodné prezentaci dat získaných prostřednictvím controllerů a také zobrazení formulářů a následnému zaslání dat z formulářů do controllerů. Zde jsou využity šablony Blade (viz kapitola 3), které umožňují spolehlivou a snadnou spolupráci PHP a HTML. Jedná se tedy o soubory zobrazující a strukturující webové stránky pomocí jazyka HTML, kterému jsou doplněny třídy frameworku Bootstrap, jež doplňují vzhled stránek. Prezentace dat je realizována pomocí jazyka PHP. Například prezentace seznamu všech projektů v systému je uskutečněna cyklem právě v jazyce PHP, který je díky šalboně Blade možno vložit přímo do HTML kódu. Pokud se jedná o šablonu obsahující formulář, nebo nějaké další odkazy, pak je na příslušné routy vždy odkazováno pomocí jejich názvu, jak již bylo uvedeno výše.

6.8 Vícejazyčnost

Informační systém pro řízení projektů v IT firmě podporuje český, anglický a slovenský jazyk. Uživatel může změnit jazyk systému v pravém horním rohu hlavní lišty v horní části okna. Zde je umístěna rozbalovací nabídka (dropdown), kde je napsána zkratka právě používaného jazyka a šipka, která rozbalí nabídku ostatních jazyků. Jazyky jsou zde uvedeny pomocí zkratek, tedy CZ, EN a SK. Při kliknutí na některý jiný jazyk než je aktuální jazyk systému přejde k přesměrování na jazykovou routu. Tato routa obsahuje funkci, která uloží do session (sezení/spojení mezi klientem a serverem) zkratku jazyka, do kterého má být systém přeložen. Další důležitou komponentou je middleware právě pro nastavování jazyků. Tato middleware se jmenuje Language a stejně jako ostatní middlewary je zaregistrována v app/Http/Kernel.php V této middleware je funkce handle, která provádí nastavení jazyka systému podle toho, co je právě uloženo v session. Pokud v session nic není, nastavuje jazyk systému implicitně podle konfiguračního souboru.

V pohledech (views), kde se nevypisují žádné proměnné, ale napevno daný text, jako například pojmenování tlačítek je nutné, aby byl tento text překládán. Proto jsou všechny takové texty nahrazeny odkazy (trans a cesta] na jazykové soubory. Každý jazykový soubor pak existuje ve třech variacích, pro každý jazyk. Laravel automaticky podle právě nastaveného jazyka systému určí, jaký jazykový soubor bude aktuálně sloužit jako zdroj textových dat.

6.9 Model

Model slouží v architektuře MVC (Model View Controller) k práci s daty (získávání dat, ukládání dat a tak dále). Nezáleží na tom, zda jsou data uložena v databázi, nebo třeba v souborech na disku. Právě kvůli zmíněné skutečnosti je aktuální sekce a část implementace systému pojmenována modely. Pro potřeby této práce jsou všechna data uchována v databázi a pomocí modelů jsou získávána, zpracována, nebo ukládána. K tomu slouží třídy repozitáře (repositories) pojmenované podle návrhového vzoru Repository. Každá třída spravuje data jedné části systému, jako například data o projektech, týmech, uživatelích a podobně. Repozitáře obsahují metody, jako třeba pro získání všech uživatelů, získání jednoho uživatele, vložení dat o novém uživateli do databáze, aktualizace stávajícího záznamu a tak podobně. Tyto metody jsou volány z controllerů, kam jsou také vráceny zpracované výsledky vyhledávání v databázi.

V metodách repozitářů je pro práci s databází použit nástroj Query Builder frameworku Laravel. Pomocí tohoto nástroje mohou být utvářeny různé typy SQL dotazů. Po získání dat z databáze je vytvořen nový objekt a tato data jsou do něj vložena. Hodnota každého sloupce databáze je vložena do samostatného atributu třídy. Framework Laravel k tomuto využívá třídy Eloquent, obsahující mnoho závislostí na ostatních třídách rozšiřujících jejich možnosti a funkcionalitu. Nicméně pro běžné informační systémy je toto nepotřebné a zbytečně to zpomaluje běh systému.

V této práci byly implementovány jak Eloquenty, tak jejich alternativa. Alternativou pro Eloquenty vytvořenou speciálně pro tuto bakalářskou práci jsou jednoduché třídy (itemy) obsahující atributy pro všechny sloupce databázové tabulky, jejíž hodnotami budou naplněny. Tyto třídy obsahují dále metody pro získání hodnot atributů, takzvané gettery (getters) a několik dalších pomocných metod. Při vytvoření instancí těchto tříd jsou do konstruktoru předána hrubá data získaná pomocí Query Builderu z databáze. Tato data jsou v konstruktoru zpracována a přiřazena do atributů třídy. Třída tedy neobsahuje žádné další závislosti a měla by tedy být rychlejší variantou pro získávání a zpacování dat z databáze. Podrobněji je tato problematika popsána v kapitole 8 - Měření, kde probíhá měření času odezvy systému pro obě varianty a následné vyhodnocení. V případě Eloquentů jsou data z databáze také získávána pomocí Query Builderu. Nicméně je k tomu využíváno právě Eloquentů, do kterých jsou data následně vložena. Eloqenty i jejich alternativa implementují stejné rozhraní, tedy obsahují stejné metody (gettery) a mohou tak být bez problému použity v controllerech či později v pohledech, aniž by musel být brán zřetel na to, která varianta je právě využívána. Pro každou variantu je vytvořena sada repozitářů. Obě sady repozitářů však implemntují opět stejné rozhraní. Uživatel si může vybrat jakou variantu bude používat, a to prostřednictvím přepínání driverů v horní liště informačního systému.

6.10 Helpery

Helpery jsou v této práci využity k usnadnění přístupu k metodám repozitářů z controllerů. Jedná se o seznam funkcí vložených do PMSServiceProvideru. Pro každý repozitář existuje jedna funkce, která vytvoří instanci třídy daného repozitáře a vrátí ji jako návratovou hodnotu funkce. Z controllerů jsou pak tyto funkce volány a prostřednictvím vráceného objektu je pak možné přistupovat k metodám daného repozitáře. Instance repozitáře je vytvářena prostřednictvím jeho rozhraní (interface), jelikož není předem jisté, zda bude použit repozitář využívající Eloquenty nebo jejich alternativu (Itemy). Mapování rozhraní

na konkrétní repozitáře je implementováno v PMSServiceProvideru a závisí na aktuálně používaném driveru.

Příkad helperu pro repozitář spravující data týkající se souborů (FileRepository) je na obrázku **6.8**.

```php
if (! function_exists('fileRepository')) {
    function fileRepository() : \xkamen06\pms\Model\FileRepositoryInterface
    {
        static $repo = null;
        if (null === $repo) {
            $repo = app(xkamen06\pms\Model\FileRepositoryInterface::class);
        }
        return $repo;
    }
}
```

Obrázek 6.5: Příklad helperu pro třídu FileRepository

6.11 Mezipamět (cache)

Pokud je potřeba získat nějaká data z databáze, jsou z controllerů prostřednictvím helperů volány metody repozitářů. Ty získají potřebná data a ve vhodné formě, například v podobě objektů, je vrátí zpět do příslušného controlleru, který tato data vloží do pohledů, kde jsou prezentována uživateli. Pokud však mnoho uživatelů velmi často žádá zobrazení stejných dat, například seznam všech uživatelů systému, znamená to spoustu dotazů do databáze. Přestože mohou jednotlivé dotazy trvat zanedbatelný čas, jejich časté opakování tento čas zvyšuje. Proto je v repozitářích implementováno využití cache, neboli mezipaměti. Načtená data z databáze jsou označena značkou (cacheTags) a pod určitým klíčem (cacheKey) jsou vložena do mezipaměti. Zde jsou tak dlouho, dokud nevyprší jejich platnost. Ta může vypršet po uplynutí doby, po kterou mají být data uchována (zadáno v repozitáři), nebo při jejich změně. Při příštím dotazování do databáze se nejprve zjistí, zda jsou daná data uložena v mezipaměti. Pokud ano, jsou z ní získána, pokud ne, proběhne dotazování do databáze.

Příklad použití funkce repozitáře, která využívá ukládaní do mezipaměti (cache) je na obrázku **6.9**.

```php
/** Get comment by id ...*/
public function getCommentById(int $commentId) : CommentInterface
{
    $cacheKey = 'CommentRepository.getCommentById' . $commentId;

    return $this->cache($this->cacheTags, $cacheKey, $this->cacheInterval, function () use ($commentId) {
        return $this->toItem($this->getQuery()->where('commentid', '=', $commentId)->first());
    });
}
```

Obrázek 6.6: Metoda využívající ukládání do mezipaměti

6.12 Notifikace

Jak je uvedeno v analýze a specifikaci požadavků (kapitola 4), tak po některých úkonech vykonaných uživatelem v systému dochází k odeslání oznámení (notifikací) jiným uživatelům. Zmiňovanou akcí může být například přiřazení úkolu. Pak je odesláno oznámení právě přiřazenému uživateli. Notifikace jsou záslány elektronickou poštou na příslušnou emailovou adresu.

Zasílání notifikací je implementováno v repozitářích. Stačí mít objekt naplněný daty o uživateli z databáze. Nad tímto objektem je poté možné zavolat metodu notify a jako parametr předat konkrétní notifikaci. Takto jednoduchým způsobem je možné zasílat oznámení proto, že Eloquent pro uživatele, nebo jeho alternativa využívají traitu frameworku Laravel pro notifikace. Šablony notifikací, tedy to, jak bude výsledný email vypadat, jsou uloženy v adresáři Notifications v kořenovém adresáři balíčku s informačním systémem.

Kapitola 7

Testování

Na následujících řádcích bude stručně popsáno testování systému. Jeho hlavní a nejspíše nejvíce podstatná část, kterou tvoří repozitáře a eloquenty a jejich alternativa (itemy), byla testována pomocí jednotkových (unit) testů podporovaných frameworkem Laravel. Pro každý jeden repozitář byla vytvořena třída obsahující jeho jednotkové testy. Tyto třídy rozšiřují testovací třídu TestCase frameworku Laravel, a tím pádem mohou využívat různé užitečné metody této třídy, jako například metody, které zjistí zda dané pole není prázdné, zda určitá proměnná je nastavena na null a další. Jedná se o assert metody a jsou volány z metod nově vytvořených testovacích tříd. Testovací třídy obsahují metodu, která získá z databáze jeden objekt, související s právě testovaným repozitářem, eloquentem a itemem. Tento objekt je následně použit při testování. Na obrázku **7.1** je možné pro názornost vidět dva testy, testující jednu a tu samou metodu repozitáře. Jedná se o metodu, která vrací objekt pro komentář daný identifikátorem - id. Pokud je identifikátor platný a daný záznam s tímto identifikátorem v databázi existuje, pak je vrácen, buď jako objekt eloquent, nebo jako objekt item (alternativa eloquentu). To záleží na právě používaném driveru. Pokud identifikátor platný není, je vyhozena vyjímka. Všechny testy jsou plně automatizované a je možné je spouštět v programu PhpStorm (IDE pro jazyk PHP) [11] po jednom, po třídách, nebo všechny naráz. Po ukončení testů je databáze vrácena do stavu před testováním. Všechny metody repozitářů obstály v jednotkových testech a lze je tedy považovat z plně funkční a bez problému použitelné v této bakalářské práci.

```php
/** @test */
public function get_comment_by_id_with_existing_id_should_return_object()
{
    $commentId = $this->getTestCommentObject('article')->commentid;
    $comment = commentRepository()->getCommentById($commentId);
    $this->assertEquals($commentId, $comment->getCommentId());
}

/** @test */
public function get_comment_by_id_without_existing_id_should_throw_exception()
{
    $this->expectException(NotFoundHttpException::class);
    commentRepository()->getCommentById(0);
}
```

Obrázek 7.1: Příklad unit testů

Kapitola 8

Měření

V této kapitole budou velmi stručně popsány obě již zmíněné varianty získávání dat z databáze a jejich zpracování. Následně proběhne měření a zaznamenání časů odezvy webových stránek informačního systému pro každou z variant. Měření proběhne pro několik stránek s různými daty získanými z databáze. Tyto stránky budou následující:

- stránka pro zobrazení všech uživatelů systému,
- stránka pro zobrazení všech projektů a
- stránka s detailem vybraného projektu.

U uvedených webových stránek systému budou měřeny hodnoty jejich odezvy v závislosti na právě zvoleném získávání dat z databáze. Z těchto získaných hodnot bude vybrán vzorek, který vznikne odebráním několika nejnižších a nejvyšších hodnot. Z výsledného vzorku budou vypočítány aritmetické průměry a bude určeno kolik času se v průměru ušetří při používání alternativy Eloquentů oproti používání samotných Eloquentů. Dále budou také vygenerovány názorné grafy prezentující rozdíly právě mezi těmito dvěma variantami získávání a zpracování dat. Následně bude uvedeno, jaký vliv má ukládání dat do mezipaměti (cache) na čas odezvy výše uvedených webových stránek informačního systému pro řízení projektů v IT firmě.

Velká část vývojářů webových informačních systémů, redakčních systémů a eshopů využívá pro jejich tvorbu framework Laravel. Většina z nich také využívá implicitní získávání dat z databáze a následné zpracování pomocí tříd Eloquentů. Data jsou z databáze získána pomocí nástroje Query Builder a následně ukládána do předem připravených tříd Eloquentů. S těmito naplněnými třídami se dále pracuje v controllerech a pohledech (view). Pro vývojáře je tato varianta poměrně komfortní, jelikož jsou data získáná pomocí Query Builderu automaticky vkládána do jednotlivých atributů Eloquentů. Přístup k těmto datům je poté velmi triviální. Buď se přistupuje rovnou k atributům, nebo jsou hodnoty atributů získány pomocí k tomu určených metod (getterů). Toto řešení není však příliš rychlé. Webové aplikace částěčně zpomaluje fakt, že třídy Eloquentů obsahují velké množství závislostí a rozšíření, které pro běžné projekty nejsou potřebné.

V této bakalářské práci byla navržena a implementována alternativa Eloquentů ve snaze zrychlit proces získání a zpracování dat z databáze. Tato alternativa byla pojmenována Itemy. Jedná se o třídy, které neobsahují žádné nepotřebné závislosti či nadbytečné funkcionality. Obsahují pouze atributy odpovídající jednotlivým hodnotám získaným z databáze a metody, které hodnoty těchto atributů poskytují (gettery). Data jsou z databáze získána

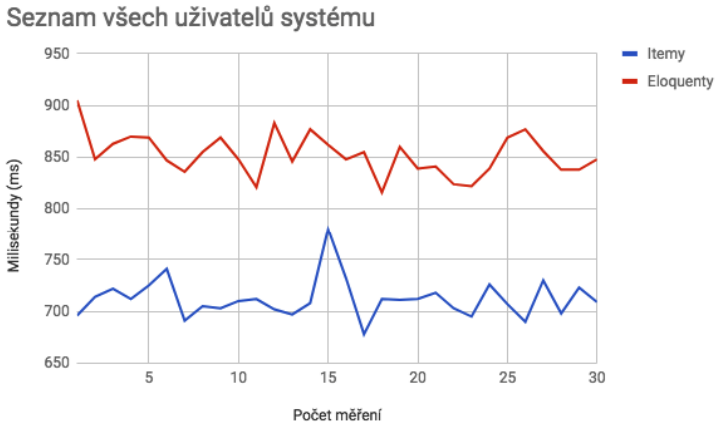
opět pomocí Query Builderu a poté jsou předána do konstruktorů nově vytvořeným objektům - Itemům. V konstruktorech jsou data přidělena do jednotlivých atributů příslušného Itemu. Odtud jsou data získána pomocí getterů a využívána opět v controllerech a pohledech. Tato varianta by měla být rychlejší hlavně z hlediska toho, že nenačítá žádné závislosti a nepracuje s žádnými jinými objekty.

8.1 Měření a získávání dat

Pro každou z výše uvedených stránek webového informačního systému bylo provedeno 40 měření pro obě varianty získávání a zpracování dat. To znamená, že každá stránka byla 40x načtena v prohlížeči pro každou variantu a všechny hodnoty byly zaznamenány do tabulky. Hodnoty byly získávány pomocí nástroje Laravel Debugbar, podrobněji popsaného v kapitole 3 - Použité technologie. Z těchto zaznamenaných hodnot bylo vždy odstraněno 5 nejnižších a 5 nejvyšších proto, aby nedocházelo ke zkreslení výsledků. Jak je možné vidět v následujících odstavcích a grafech, tak naměřené hodnoty více či méně kolísají. To je dáno připojením k internetu a dalšími závislostmi webového informačního systému pro řízení projektů v IT firmě a samotného frameworku Laravel. Nicméně je možné vidět trend, který ukazuje, že nově vytvořená alternativa je podstatně rychlejší než řešení využívající Eloquenty. Ze získaných dat byly vypočítány aritmetické průměry a rozdílem průměrů časů při využívání Eloquentů a Itemů byl vypočítán ušetřený čas při používání druhé zmíněné alternativy.

8.2 Stránka zobrazující všechny uživatele systému

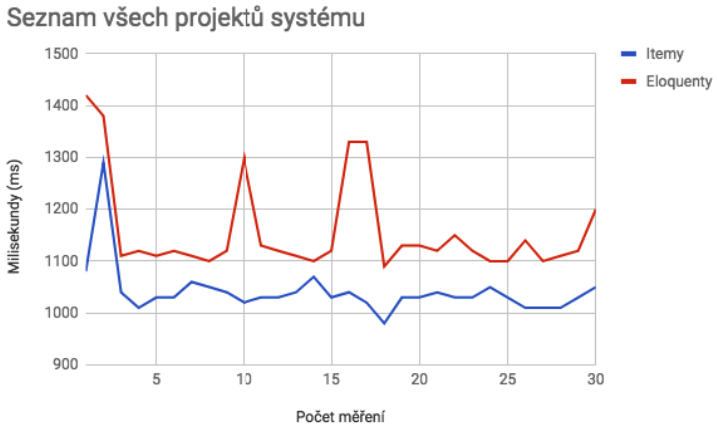
Jako první byla testována stránka pro zobrazení všech uživatelů systému. Pro názornost bylo zobrazováno 400 uživatelů systému na jednu stránku. Průměrný čas odezvy stránky při využití Eloquentů činil **852.3 ms (miliseknud)**, naopak při použití Itemů (alternativy k Eloquentům) činil průměrný čas odezvy **712.0 ms**, což znamená, že při načítání seznamu uživatelů čítajícího 400 záznamů je při použití Itemů doba získání dat kratší o **140.3 ms**. Počet dotazů do databáze přitom zůstavá stejný. Čím vyšší počet záznamů, tím se ušetřený čas zvyšuje. Graf naměřených hodnot pro obě varianty se nachází na obrázku **8.1**.



Obrázek 8.1: Graf odezvy stránky zobrazujíci všechny uživatele systému pro obě varianty

8.3 Stránka zobrazující všechny projekty

Jako další byla vybrána stránka informačního systému zobrazující projekty. Pro účely měření bylo nastaveno 100 projektů na stránku. Průměrný čas odezvy stránky při použití Eloquentů byl **1158.0 ms** tedy **1.158s**. Při použití Itemů byla na stejné stránce při stejném počtu objektů a dotazů do databáze naměřena odezva **1041.0 ms** tedy **1.041s**, což je o **117.0 ms** nižší čas odezvy, než při použití Eloquentů. Graf znázorňující toto měření je na obrázku **8.2**.



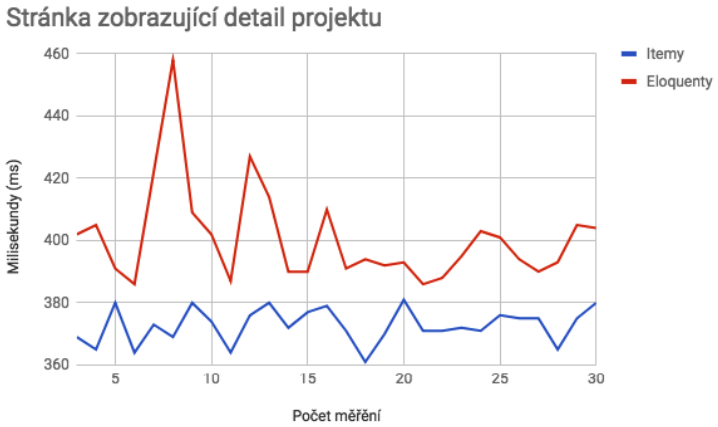
Obrázek 8.2: Graf odezvy stránky zobrazující všechny projekty systému pro obě varianty

8.4 Stránka zobrazující detail vybraného projektu

Pro toto měření byl vybrán projekt obsaující 11 úkolů, 7 souborů, a 5 řešitelů. Jedná se tedy o projekt menšího nebo středního rozsahu. Čas odezvy stránky zobrazující tento detail projektu za použití Eloquentů činil **400.5 ms**, přičemž čas odezvy při použití Itemů **372.0 ms**. I při né přílíš vysokém počtu načítaných objektů se při stejném počtu dotazů do databáze snížil čas odezvy při použití Itemů průměrně o **28.0 ms**. Názorný graf je na obrázku **8.3**.

8.5 Úkládání získaných dat z databáze do mezipaměti (cache)

V systému bylo implementováno také ukládání získaných a zpracovaných dat z databáze do mezipaměti (cache). Díky tomu jsou takto získaná a zpracovaná data uložena do mezipaměti na určitou dobu. Pokud je na tato data z controllerů po dobu jejich uchování v mezipaměti dotázáno, pak jsou vrácena právě z mezipaměti. Neprobíhá tedy žádné dotazování do databáze a plnění Eloquentů nebo Itemů. To opět šetří čas systému. Do databáze se znovu dotazuje při vypršení času, po který byla data uložena v mezipaměti, nebo pokud jsou data změněna. Měření odezvy stejných stránek, jako v předchozích odstavcích (kromě stránky s detailem projektu) bylo provedeno také při použití mezipaměti a výsledky je možné vidět v následující tabulce (**8.3**).



Obrázek 8.3: Graf odezvy stránky zobrazujíci detail projektu pro obě varianty

	Seznam uživatelů		Seznam projektů	
	Bez mezipaměti	S mezipamětí	Bez mezipaměti	S mezipamětí
Eloquenty	852 ms	622 ms	1158 ms	566 ms
Itemy	712 ms		1041 ms	

Obrázek 8.4: Tabulka s odezvami stránek systému při použití Eloquentů, Itemů a mezipaměti

Kapitola 9

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat informační systém pro řízení projektů v IT firmě. Důraz byl kladen na rychlost získávání dat z databáze a jejich zpracování. V této technické zprávě byly nejprve popsány základní pojmy týkající se informací, databází, objektově orientovaného programování, systémů, informačních systémů a dalších. Následně byly představeny použité technologie při tvorbě tohoto informačního systému včetně jazyka PHP, HTML, MySQL, UML a tak podobně. Popsán byl také PHP framework Laravel, ve kterém je celý systém vytvořen. Nebylo opomenuto ani uvedení důležitých použitých nástrojů Laravelu včetně například Query Builderu a Eloquentu. Dále byla provedena analýza s specifikace požadavků a její výsledky podrobně sepsány v příslušné kapitole. Na analýzu a specifikaci následně navázal samotný návrh systému včetně vytvoření přiložených Use Case diagramů a jednoho ER diagramu. Poté byla představena implementace jednotlivých částí informačního systému a byly uvedeny konkrétní použité technologie a techniky, jako například využití architektury MVC (včetně vysvětlení použití jejich komponent - Model, View, Controller), která je nedílnou součástí frameworku Laravel. Následně bylo stručně popsáno testování systému, konkrétně jeho částí pracující s daty, pomocí jednotkových testů.

Nejdůležitější částí bylo vytvoření vhodné alternativy k již existujícímu získávání a zpracování dat, které je běžně používáno pro aplikace vytvořené ve frameworku Laravel. Stručně řečeno byly využívané Eloquenty nahrazeny rychlejším řešením v podobě Itemů. V dalším kroku bylo nutné změřit, zda je toto řešní opravdu rychlejší. Měření probíhalo na několika různých typech stránek systému pomocí nástroje Laravel Debugbar. Hodnoty byly zaznamenávány a dále zpracovány. Při tomto procesu byl učiněn závěr, že je nově vytvořená alternativa Eloquentů opravdu rychlejší, aniž by ztratila na funkčnosti podstatné pro vytváření běžných webových aplikací. Bylo také zjištěno, že čím více dat systém obsahuje a zpracovává, tím je řešení s Itemy rychlejší oproti původnímu řešení s Eloquenty.

Z hlediska samotného informačního systému jsou další možná rozšíření spíše otázkou reálného nasazení systému a následných dalších požadavků potenciálního klienta. Každá IT firma může řídit své projekty trochu jiným způsobem, a proto by pak zaléželo na konkrétních potřebách dané firmy. Nicméně funkční široký základ informačního systému pro řízení projektů v IT firmě byl vytvořen v rámci této bakalářské práce a je schopen plného nasazení. Z hlediska dalšího zrychlování systému je možné se zamyslet, zda některá z dalších komponent architektury Laravel neobsahuje přílišné závislosti a nezpomaluje tak systém. Jednou z možných komponent může být samotný Query Builder.

Literatura

[1] *Bitbucket* . [Online; navštíveno 02.04.2018].
URL https://bitbucket.org/

[2] *Bootstrap The most popular HTML, CSS, and JS library in the world.* [Online; navštíveno 28.03.2018].
URL https://getbootstrap.com/

[3] *Composer.* [Online; navštíveno 28.03.2018].
URL https://getcomposer.org/

[4] *Computer Hope - What is HTML (HyperText Markup Language) ?* . [Online; navštíveno 02.04.2018].
URL https://www.computerhope.com/jargon/h/html.htm

[5] *IT Network.* [Online; navštíveno 28.03.2018].
URL https://www.itnetwork.cz/

[6] *Jak psát web, CSS - kaskádové styly.* [Online; navštíveno 28.03.2018].
URL https://www.jakpsatweb.cz/css/

[7] *Laravelblog Vše o frameworku Laravel.* [Online; navštíveno 28.03.2018].
URL https://www.laravelblog.cz/

[8] *Laravel - The PHP Framework For Web Artisans.* [Online; navštíveno 28.03.2018].
URL https://laravel.com/

[9] *npm - package manager.* [Online; navštíveno 28.03.2018].
URL https://www.npmjs.com/

[10] *PHP: Hypertext Preprocessor.* [Online; navštíveno 28.03.2018].
URL http://www.php.net

[11] *PhpStorm: Lightning-Smart IDE for PHP Programming by JetBrains* . [Online; navštíveno 01.05.2018].
URL https://www.jetbrains.com/phpstorm/

[12] *Sass: Syntactically awesome style sheets.* [Online; navštíveno 28.03.2018].
URL https://sass-lang.com/

[13] *W3Schools - CSS tutorial.* [Online; navštíveno 28.03.2018].
URL https://www.w3schools.com/css/

[14] Barry, H.: *GitHub - barryvdh/laravel-debugbar: Laravel Debugbar (Integrates PHP Debug Bar)* . [Online; navštíveno 28.03.2018].
URL https://github.com/barryvdh/laravel-debugbar

[15] Hruška, T.; Burget, R.: *Internetové aplikace (WAP) II. část SGML, HTML, CSS, DOM Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 22.02.2018].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/podklady/Opory/OWAPSGML.pdf

[16] Hruška, T.; Burget, R.: *Internetové aplikace (WAP) IV. část Programování serveru (PHP) Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 22.02.2018].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/podklady/Opory/OWAPPHP.pdf

[17] Hruška, T.; Křivka, Z.: *Informační systémy Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 27.03.2018].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/podklady/Opory/OIISPojem.pdf

[18] Křivka, Z.; Kolář, D.: *Principy programovacích jazyků a objektově orientované programování Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 17.04.2018].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIPP-IT%2Ftexts%2FIPP-II-ESF-1_2_hyperlinks.pdf&cid=11469

[19] Welling, L.; Thomson, L.: *PHP a MySQL: rozvoj webových aplikací*. SoftPress, 2003, ISBN 80-86497-60-7.

[20] Zaninotto, F.: *GitHub - fzaninotto/Faker*. [Online; navštíveno 26.04.2018].
URL https://github.com/fzaninotto/Faker

[21] Zemčík, P.: *Tvorba uživatelských rozhraní Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 20.03.2018].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FITU-IT%2Ftexts%2FlTU-Podpora.pdf&cid=12203

[22] Zendulka, J.; Rudolfová, I.: *Databázové systémy Studijní opora – Fakulta informačních technologií VUT v Brně*. [Online; navštíveno 27.03.2018].
URL
https://www.fit.vutbr.cz/study/courses/IDS/private/opora/IDS_predn.pdf